HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ipgag
THE EXPERTS IN IAM

Rollenmanagement Tool

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2013

Autor(en): Florian Mahler, Marco Sonderegger
Betreuer: Hans Noser
Projektpartner: www.ipg-ag.com, 8406 Winterthur
Experte: Daniel Reisacher
Gegenleser: -

Abstract

In unserer Studienarbeit haben wir ein Rollenmanagement Tool, im weiteren RMT genannt entwickelt.
RMT ist eine webbasierte Lösung, um Rollen zu verwalten, zu erstellen und zu versionisieren.
Man kann mit diesem Webtool neue Rollen mit den benötigten Rechten erstellen, bearbeiten. Die Rollen können mit all ihren Versionen angezeigt werden. Mit RMT kann man neue Rollenversionen erstellen und bearbeiten. Es können auch ältere Rollenversionen verwendet werden, um neue Rollen oder neue Versionen von bestehenden Rollen zu erstellen.
Da es im Rollenmanagement Tool nur um die Verwaltung und Versionisierung von Rollen geht, werden all benötigten Daten, welche nicht manuell bei der Erfassung der Rollen bzw. Rollenversionen angegeben werden, in das System geladen. Dies geschieht mittels eines Imports Prozess.
Die Import Funktion in RMT funktioniert mit .csv Dateien. Es können .csv-Dateien mit den Rechten und den dazu benötigten Daten, wie zum Beispiel zu welcher Applikation ein Recht gehört, importiert werden. Auch Rollen können via .csv-Dateien importiert werden.
Neben der Import Funktion gibt es auch eine Export Funktion, welche vorhandenen Daten, also die Rollen ihren Versionen und den Rechten, usw., .csv-Dateien exportieren kann.
RMT ist ein mit JSF und Java entwickelt worden, um möglichst auf vielen Server Systemen betrieben zu werden. Auch können ziemlich viele Datenbanken mit RMT benützt werden. Denn es muss nur die Konfiguration entsprechend angepasst werden. Denn in RMT benützen wir JPA für das Datenbank Management, was die Interoperabilität gewährleistet.

Inhalt

Contents

Management Summary

Änderungsgeschichte

Datum	Version	Änderung	Autor
30.05.2013	1.0	Initialversion	msondere

1. Ausgangslage

Zum jetzigen Zeitpunkt gibt es keine Tools auf dem Markt, mit dem man Rollen verwalten und versionisieren kann. Es gibt auch in keinem Programm die Möglichkeit anhand von den älteren Rollenversionen neue Rollen zu bilden, ohne dass die alte Rolle bestehen bleibt und weiterhin zu Verfügung steht. Somit kann man nicht nachvollziehen, wie es zu den entsprechenden Rollen gekommen ist und welcher Gedanke dahinter stand. Man kann auch nicht einsehen zu welcher Zeit eine Rolle im Einsatz, also aktiv war. Zudem soll das zu entwickelnde Tool mit gängigen Rolmining- und Provisionierungstools kommunizieren können. Es sollen vorhandene Rollen und Rechte von vorhing genannten Tools mittels Schnittstelle, einem Importprozess, in das zu entwickelnde System eingespielt werden. Natürlich soll es auch einen Exportprozess geben, der die vorhandenen Daten im Rollenmanagement-Tool exportiert. Diese Exportdateien können dann in ein Identity and Access Management (IAM) oder auch (IdM) genannte Systeme importiert werden.

Genau dies soll mit dem zu entwickelnden Rollenmanagement-Tool, im weiteren Verlauf des Dokuments auch als RMT bezeichnet, erzielt werden. RMT soll als webbasierte Lösung entwickelt werden und soll zu einem späteren Zeitpunkt weiterentwickelt und durch zusätzliche Funktionen ergänzt werden.

2. Vorgehen, Technologien

Um das Rollenmanagement Tool zu entwickeln, standen uns entweder C# / .Net oder Java zu Verfügung. Deshalb haben wir zuerst diese beiden Programmiersprachen gegeneinander verglichen. Nach folgend ist der Vergleich in einer Tabelle dargestellt. Dieser Entscheid war sehr wichtig, denn es müssten je nach Wahl, ganz andere Technologien verwendet werden und selbstverständlich würde sich die Architektur auch ziemlich fest verändern.

		Java		**C#**	
Vergleichs-position	Gewichtung	*Beschreibung*	*Note*	*Beschreibung*	*Note*

Plattform-unabhängigkeit	2	Plattformunabhängig, Code kann ohne Veränderungen oder Einschränkungen auf allen Systemen, für welche es eine JVM gibt, ausgeführt werden	10	Grundsätzlich plattformunabhängig unter Verwendung von mono, jedoch gewisse Kompatibilitätsprobleme auf nicht-MS-Plattformen	5
Performance	3	Gute Performance auf allen Systemen	8	Sehr gute Performance auf MS-Systemen, falls kein mono verwendet wird (kompletter Verzicht auf Plattformunabhängigkeit!), ansonsten mässige Performance	6
DB Zugriff	2	JPA: Hibernate / EclipseLink, beides starke Frameworks, beide OpenSource	9	Nhibernate, OpenSource, gutes Framework, aber nur wia .xml konfigurierbar	6
DB Flexibilität	2	DB leicht austauschbar durch Änderung der Konfiguration	9	DB austauschbar durch Änderung des ConnectionStrings, welcher ausgelagert werden kann	8
Webservice-Plattform	2	Alle Webserver, grosse Auswahl an Servletcontainer: Tomcat, Glassfish, Jboss, WebSphere, …	10	Grundsätzlich nur IIS, Unterstützung für andere Webserver nur vereinzelt und nur für ältere Technologien, vielfach ohne Support oder Community	6
Webtechnologie	2	JSF oder JSP, stabile, sichere und weit verbreitete Technologien, jedoch etwas aufwändig zum entwickeln	7	asp.net oder asp.net MVC, wobei letzteres nicht sicher Plattformunabhängigkeit bieten kann	7
Know-How des Projektteams	1	sehr gut	9	mässig	4
Wunsch des Industrie-partners	5	Wenn es denn sein muss…	3	Sehr gerne!	10
Gewichtungs-summe:	19	Durchschnittsnote:	7.263157895		7.157894737

Vergleich Java vs. C# / .Net

Benotung von 1 bis 10,
Gewichtung von 1 bis 5

Da vom Auftraggeber C#/.Net gewünscht wurde haben wird, dies extrem im Vergleich der beiden Sprachen berücksichtigt. Wie der Tabelle zu entnehmen ist, hat Java, wenn auch nur knapp, das Rennen gemacht. Der Grund für diese Entscheidung ist einerseits das Ergebnis des Vergleichs und andererseits das „Know-How“ im Bereich Java des Teams. Im Endeffekt hat unser Betreuer Herr

Noser uns die Erlaubnis erteilt Java zu verwenden aus den vorhin genannten Gründen, obwohl C#/.Net bevorzugt worden wäre.
Da RMT eine webbasierte Lösung werden sollte, wählten wird für das Frontend „JSF“ mit „.xhtml“-Webseiten. Ein weiterer Grund dafür ist, dass es für „JSF“ zwei sehr nützliche Frameworks gibt, mit denen die Benutzeransicht im Browser viel eleganter und einfacher gestaltet werden kann.
Eines dieser Frameworks, welches wir verwenden nennt sich „Primefaces“.
Dazu kommt, das diese Frameworks uns viele Probleme abnehmen, welche wir sonst zusätzlich lösen hätten müssen.
Für den Backend Teil unserer Applikation, also der Teil der Applikation, der nur auf einem Applikationsserver läuft, haben wir uns für „J2EE“ und ihren „EJB3“ (Enterprise Java Bean) und „JPA“ (Java Persistence API) entschieden.
Die „EJB“ fungieren als Kontroller, welche Zugriff auf die Daten via, einem zu Verfügung gestellten Service geben.
JPA wird gebraucht, um die Domainklassen so zu konfigurieren, dass die erstellten Objekte der Klassen, mittels dem zu Verfügung gestelltem Service, der „EJB3“, via Benutzer Interaktion in eine Datenbank geschrieben werden.
Als Applikationsserver, in dem RMT läuft, haben wir und für JBoss entschieden. Für die Datenbank verwenden wir MySQL.
Dies ist jedoch kein Muss für RMT. Es kann in einem beliebigen Applikationsserver mit einer beliebigen Datenbank betrieben werden.

Auch wenn diese Technologien uns viel Zeit und Arbeit erspart haben, hatten wir so unsere Probleme. Da wir nicht alle dieser Technologien gut kannten, brauchten wir eine gewisse Zeit, um uns dieses Wissen anzueignen.
Wir haben während unserer Semesterarbeit herausgefunden, dass die verwendeten Technologien zwar ihre Vorzüge haben, jedoch auch kryptische Fehlermeldungen hervorbringen, mit denen man einfach nichts anfangen kann. Die einzige Möglichkeit war in diesem Fall, mittels www.google.ch danach zu suchen, um das Problem zu eruieren. So haben wir viel Erfahrung gesammelt und konnten, weitere ähnliche Probleme schneller lösen.

3. Ergebnisse

Das Rollenmanagement Tool ist schlussendlich lauffähig.
Es sind fast alle der geforderten Funktionen, welche in den Anforderungsspezifikationen bei den funktionalen Anforderungen beschrieben wurde, vorhanden.
Die einzige Ausnahme sind die Mandanten-Lösung, sowie das gewünschte Login. Diese beiden Funktionen konnten aus zeitlich bedingten Gründen nicht mehr entwickelt werden.
Man kann mit RMT Rollen und Rollenversionen erstellen, kopieren und bearbeiten. Es können auch ältere Rollenversionen einer Rolle geladen werden.
Des Weiteren können die Details von Rollen und ihren Rollenversionen, sowie die Detail der Rechte, welche nach Applikationen gegliedert werden. Es können Rollentypen hinzugefügt, bearbeitet und angezeigt werden. Diese werden für die Rollen benötigt.
Zu guter Letzt, funktioniert auch der Import von Rechten und Rollen via .CSV-Datei, sowie deren Export. Die letztgenannte Funktion läuft aber noch nicht zufriedenstellend.

Projektplan

Änderungsgeschichte

Datum	Version	Änderung	Autor
14.03.2013	1.0	Initialversion	msondere
31.05.2013	1.1	Verbesserungen & Vervollständigung	fmahler

Inhalt

4. Einführung

4.1 Zweck

Dieses Dokument enthält die gesamte Planung der SA.

4.2 Gültigkeitsbereich

Dieses Dokument ist gültig für die SA von Florian Mahler und Marco Sonderegger

5. Projekt Übersicht

Es handelt sich bei diesem Projekt um unsere Studienarbeit. Für diese soll ein Rollenmanagement Tool entwickelt werden, welches webfähig ist

5.1 Zweck und Ziel

Das Ziel der SA ist es diese erfolgreich abzuschliessen. Das zu entwickelnde Tool soll in einer lauffähigen Version gemäss Anforderungsspezifikation abgegeben werden.

5.2 Lieferumfang

Siehe DokuAnleitungBA_DA_SA_110907.pdf unter https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html im Kapitel Dokumentation

6. Projektorganisation

6.1 Organisationsstruktur

Name	Bereich
Florian Mahler	Domain, Backend
Marco Sonderegger	Webfrontend

6.2 Externe Schnittstellen

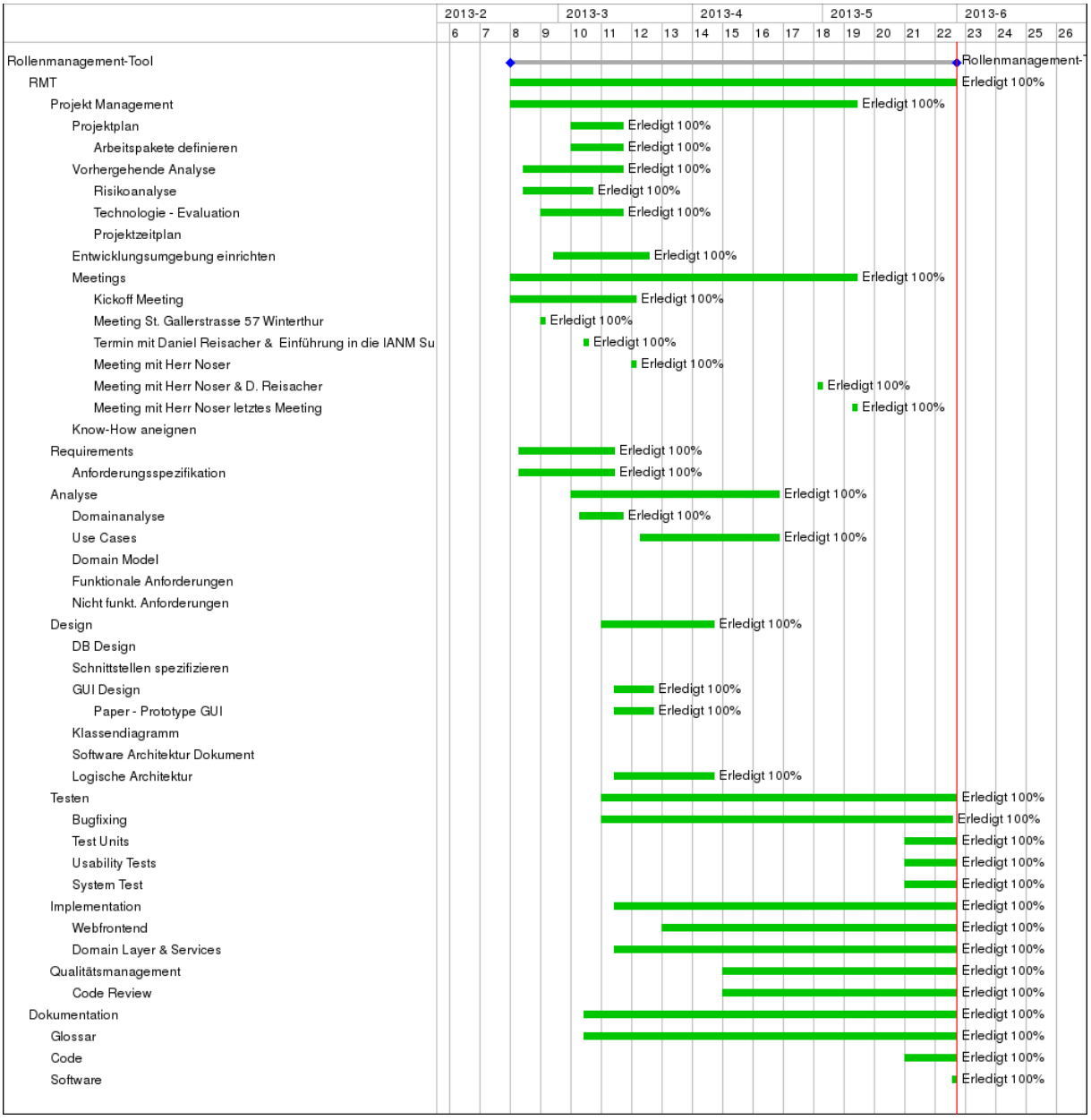
Name	Funktion
Hans Noser	Betreuer
Daniel Reisacher, IPG AG	Experte
www.ipg-ag.com	Industriepartner
tbd	Gegenleser

7. Management Abläufe

7.1 Kostenvoranschlag

Die SA dauert ein ganzes Semester à 10h die Woche. Dies entspricht 280h. Der Abgabe Termin der SA ist am 31. Mai 2013.

7.2 Zeitliche Planung



7.2.1 Phasen / Iterationen

7.2.2 Inception

Dauer der Inception Phase 2 Wochen. In dieser Phase wird erster Projektplan mit den Meilensteinen und Arbeitspaketen, sowie die erste Analyse der Aufgabenstellung der Semesterarbeit gemacht. Das Kickoff Meeting der Semesterarbeit wird abgehalten.

7.2.2.1.1 Iteration Inception 1

Dauer der 1. Inception Iteration ist eine Woche.
Anfangs dieser Iteration findet das Kickoff Meeting statt. Erster grober Zeitplan wird erstellt und die Dokumente werden aufgesetzt. Die Aufgabenstellung wird analysiert und ein erster Projektplan wird erstellt.

7.2.2.1.2 Iteration Inception 2

Dauer der 2. Inception Iteration ist ebenfalls eine Woche. Die Entwicklungsumgebung wird begonnen einzurichten. Zeitplan und Projektplan wird verfeinert. Erste Skizzen und Ideen für die Umsetzung der Aufgabenstellung werden erstellt.
Es wird mit der Risikoanalyse und der Technologie Evaluation begonnen.

7.2.3 Elaboration

Dauer der Gesamten Elaboration Phase sind sechs Wochen. Analyse der SA Aufgabe wird weiter getrieben und am Ende der drei Iterationen sollte ein Grossteil der der Analyse, Anforderungsspezifikation und dem Design stehen. Es wird natürlich auch schon mit der Entwicklung des 1. Prototyps begonnen.

7.2.3.1.1 Iteration Elaboration 1

Dauer dieser Iteration sind zwei Wochen.
Entwicklungsumgebung wird fertig eingerichtet werden. Mit der Anforderungsspezifikation und der Domainanalyse soll begonnen werden. Die Risikoanalyse und die Technologie Evaluation begonnen sollen vorangetrieben und fertig gestellt. Der Projektplan soll fertiggestellt werden.
Es soll mit Entwickeln begonnen werden.

7.2.3.1.2 Iteration Elaboration 2

Dauer der Elaboration Iteration 2 sind zwei Wochen.
Die Anforderungsspezifikation soll zu 60% fertig sein. Die Funktionalen Anforderungen sind definiert und es können nur noch minime Änderungen vorgenommen werden. Es wird mehr entwickelt als in der vorherigen Iteration. Domainmodell soll fertiggestellt werden .Risikoanalyse und Technologie Evaluation sollten auch beendet werden. Mit dem Architekturdokument soll begonnen werden.

7.2.3.1.3 Iteration Elaboration 3

Dauer der Elaboration Iteration 3 sind zwei Wochen. In dieser soll die Anforderungsspezifikation zu 80% fertig. Die Domainanalyse sollte auch fertig werden. Es soll hauptsächlich entwickelt werden. Das Architekturdokument soll zu 80% fertig gestellt werden. Meilenstein Prototyp soll erreicht werden.

7.2.4 Construction

In dieser Phase liegt der Fokus hauptsächlich bei der Entwicklung des 2. Prototyps. Es soll weniger an der Dokumentation gearbeitet werden. Dauer der Phase Construction sind 4 Wochen.

7.2.4.1.1 Iteration Construction 1

Dauer der Construction Iteration 1 sind 2 Wochen.
Es soll am 2. Prototyp von RMT gearbeitet werden.
Die Anforderungsspezifikation und das Architekturdokument sollte zu 100% fertig erstellt werden

7.2.4.1.2 Iteration Construction 2

Dauer der Construction Iteration 2 sind 2 Wochen.
Der Endspurt zum 2. Meilenstein beginnt.
Meilenstein Pre-Release soll fertig werden.

7.2.5 Transition

Dies ist die Schlussphase. Der Endspurt bis zur Abgabe der SA beginnt. RMT soll alle vereinbarten Funktionalitäten haben. Die Dokumentation wird fertiggestellt. Dauer der Phase Transition sind 2 Wochen.

7.2.5.1.1 Iteration Transition 1

Dauer der Transition Iteration 1 ist eine Woche. Der Schlussspurt bis zur SA Abgabe beginnt. Code Refactoring, Finishing und Bugfixing des Codes sollte gemacht werden.

7.2.5.1.2 Iteration Transition 2

Dauer der Transition Iteration 2 ist eine Woche.
Überarbeiten aller Dokumente, Refactoring, Finishing und Bugfixing des Codes sollte gemacht werden.
Meilenstein SA Abgabe wird erreicht.

7.2.6 Meilensteine

Datum	Meilensteine	Arbeitsprodukte
11.04.2013	Meilenstein Prototyp	1. Prototyp von RMT
30.04.2013	Meilenstein Pre-Release	2. Prototyp von RMT
31.05.2013	Meilenstein Abgabe	SA Abgabe

7.2.6.1 Meilenstein Prototyp

An diesem Meilenstein soll der erste Prototype fertiggestellt sein, um ihn Herrn Noser zu präsentieren.

7.2.6.2 Meilenstein Pre-Release

Am Meilenstein Pre-Release soll der nächste Prototyp fertiggestellt sein, um ihn Herrn Noser und Herrn Reisacher zu zeigen.

7.2.6.3 Meilenstein Abgabe

An diesem Meilenstein muss die SA abgegeben werden und die definierten Funktionen, einsehbar im Requirements Specification.docx, lauffähig sein. Alle Dokumentente müssen erstellt sein.

7.3 Besprechungen

Tag	Datum	Uhrzeit	Thema	Wo	Wer
Montag	18.02.2013	16:00-17:30	Kickoff Meeting	ipg ag Technopark-strasse 2 8406 Winterthur	Team / H. Noser
Montag	25.02.2013	16:15-18:00	Requirements	Büro H. Noser St. Galler-strasse 57 8406 Winterthur	Team / H. Noser
Donnerstag	07.03.2013	08:30-10:30	Einführung in die IAM Suite von Quest, bzw. Dell, Technologie Evaluation	ipg ag Technopark-strasse 2 8406 Winterthur	Team / H. Noser / D. Reisacher
Montag	18.03.2013	15:30-16:30	Aktueller Stand	HSR Cafeteria	Team / H. Noser
Donnerstag	11.04.2013	14:00-16:00	Aktueller Stand / Prototyp	HSR Cafeteria	Team / H. Noser
Dienstag	30.04.2013	10:00-12:00	Aktueller Stand / Prototyp / Letzte Unsicherheiten klären	ipg ag Technopark-strasse 2 8406 Winterthur	Team / H. Noser / D. Reisacher
Mittwoch	08.05.2013	16:00-17:00	Letztes Meeting vor Abgabe, Bestimmen was noch gemacht wird und was nicht	HSR Cafeteria	Team / H. Noser

8. Risikomanagement

8.1 Risiken

Siehe Risikoanalyse.xlsx

8.2 Umgang mit Risiken

Es sind Rückstellungen von 73h geplant. Dies ist in Risikoanalyse.xlsx einsehbar.
Neubeurteilungen der Risiken werden in der Mitte des Semesters durchgeführt.

9. Arbeitspakete

Die Arbeitspakete sind in Redmine einsehbar.
URL: http://152.96.56.40/redmine bzw. http://sinv-56040.edu.hsr.ch/redmine/

Paketgruppe	Arbeitspakete

Projekt Management	• Projektplan
	  ○ Arbeitstakte definieren
	• Vorhergehende Analyse
	  ○ Projektzeitplan
	  ○ Risikoanalyse
	  ○ Technologie Evaluation
	• Entwicklungsumgebung einrichten
	• Meetings
	• Know-How aneignen

Requirements	• Anforderungsspezifikation
Analyse	• Domainanalyse • Use Cases • Domain Model • Funktionale Anforderungen • Nicht funkt. Anforderungen
Design	• GUI Paper-Prototype • GUI Design • Klassendiagramm • Software Architektur Dokument • Logische Architektur • Schnittstellen spezifizieren • DB Design
Implementation	• Webfrontend • Domain Layer & Services
Test	• Test Units • Bug fixing • Usability Tests • System Test
Dokumentation	• Software • Code • Glossar
Qualitätssicherung	• Code – Reviews

10. Infrastruktur

- 1 vSphere Server

  Die Virtual Server (vServer) laufen in einem VMWare Virtual Infrastructure Cluster mit folgenden
  Leistungsmerkmalen:

  - DMZ : 4 physikalische Server mit je 2 Quad-Core 2.27 GHz Xeon CPUs, (2 x 72, 2 x 146 GB RAM)
  - INTERN: 4 physikalische Server mit je 2 Quad-Core 2.27 GHz Xeon CPUs, (2 x 72, 2 x 146 GB RAM)
  - VMWare vSphere 4.1 (ESX)

  Dies entspricht einer Cluster Performance von 146 GHz und 872 GB RAM.

  - Konfiguration des vSphere Servers

	Linux	**Windows**
Hersteller	VMWare	VMWare
CPU	1 vCPU	1 vCPU
CPU	Max. 2.27 GHz	Max. 2.27 GHz
RAM	1 GB	2 GB
Disk Drive	15 GB	40 GB
LAN Adapter	1 VMWare Flexible NIC (VMXNET3)	1 VMWare Flexible NIC (VMXNET3)
Video Adapter	VMWare SVGA II	VMWare SVGA II
Exact OS Type	Ubuntu 10.04 LTS	Windows 2008 R2 Server
File System	EXT3	NTFS

- 2 PC im SA Zimmer
- 2 Laptops
- Tools
  - EclipseJEE Juno
  - Maven 3.1
  - JBoss AS 7.1.1
  - SVN
  - Redmine
  - SQLyog
  - SchemaSpy
  - Structure101

11. Qualitätsmassnahmen

11.1 Dokumentation

Kritische Dokumente wurden im Redmine abgelegt, wo auch automatisch eine „Versionisierung“ stattfindet, sowie gewisse Dinge auch im Dropbox. Nicht zentral abgelegt wurden generierte Daten wie Klassendiagramme oder ERMs.

11.2 Projektmanagement

Für das Projektmanagement haben wir Redmine verwendet, welches uns auf der VM zur Verfügung gestellt wurde. Dieses wurde zur Planung der Tasks, Zeiterfassung und Reporting verwendet. Es befindet sich hier:
http://sinv-56040.edu.hsr.ch/redmine/login

11.3 Entwicklung

Sämtlicher Source Code wurde im SVN Repository der HSR abgelegt und versionisiert. Da sich die Verwendung von Jenkins aufgrund des Einsatzes von Maven schwierig gestaltet hätte, haben wir darauf verzichtet, obwohl wir auf der VM einen solchen zur Verfügung gehabt hätten.

11.3.1 Vorgehen

Wir haben eine Mischung von RUP und Model driven development verwendet, um uns optimal an die Lösung heranzuarbeiten. Da das Datenmodell die treibende Kraft hinter der Applikation war, stand es stets im Mittelpunkt, während wir iterativ die benötigten Komponenten darum herum entwickelt haben. Dies hat sich letztlich als einen sehr produktiven Ansatz für unser Projekt erwiesen.

11.3.2 Unit Testing

Wir haben sehr wenig Unit Tests geschrieben, aus mehreren Gründen:

- Starken Fokus auf Umfang der Lösung
- Nur für Model sinnvoll, das Model wird aber bei jedem Systemtest komplett durchgetestet
- Probleme mit Application Container Grenze beim Instanzieren der Objekte

Wie im zweiten Punkt beschrieben, haben wir aber viele Systemtests durchgeführt, welche grundsätzlich die ganze Applikation abdecken.

11.3.3 Code Reviews

Zwei Codereviews wurden mit dem externen Experten T. Wyss durchgeführt. Sie haben direkt in Anpassungen des Domain Models und des Datenmodells sowie zu Codekommentaren („FIXME“s und „TODO“s) resultiert.

11.3.4 Code Style Guidelines

Es wurden die Standardrichtlinien für Code Style von Sun verwendet. Die einzigen erwähnenswerten Anpassungen wurden bei der Zeilenlänge des codes und der Kommentare vorgenommen.

11.4 Testen

11.4.1 Systemtests

Das System wurde vor jedem Commit auf seine Lauffähigkeit getestet, bei grösseren Änderungen auf zwei verschiedenen Maschinen. Es geht klar hervor, dass das System als Ganzes funktionstüchtig ist und alle Datenconstraints eingehalten werden.

Anforderungsspezifikation

Änderungsgeschichte

Datum	Version	Änderung	Autor
05.03.2013	1.0	Initialversion	msondere
31.05.2013	1.1	Finalversion	msondere

Inhalt

12. Einführung

12.1 Beschreibung

Mittels dem Rollenmanagement Tool (RMT) sollen Rollen erfasst, verwaltet und versionisiert werden. Es soll ein Import von Rollen und Rechten ermöglichen werden, damit die von einem Rolmining Tool oder einem Provisionierungs-Tools, wie IAM bzw. IdM, gelieferten Daten ins System geladen werden können. Der Importprozess soll .CSV-Dateien entgegennehmen.
Neben dem Import soll es auch eine Export-Funktion geben, mit welcher, die in RMT vorhandenen Daten in eine .CSV-Datei exportiert werden kann.
Das Rollenmanagement Tool soll somit den Life Cycle einer Rolle inklusive ihrer Versionen und die Dokumentation des Life Cycles sicherstellen.
RMT soll entwickelt werden, da momentan kein einziges IAM Tool in der Lage ist die obengenannten Funktionen zu bieten. Dank RMT kann der Anwender in Zukunft viel Aufwand und Geld einsparen beim Entwickeln von Rollen.

12.2 Gültigkeitsbereich

Dieses Dokument ist für das zu entwickelnde Rollenmanagement Tool bestimmt. Es werden während der ganzen Dauer der Semesterarbeit (SA) noch Änderungen, als auch Erweiterungen an diesem Dokument vorgenommen.

12.3 Referenzen

01_Projektmanagement/Projektplan.docx
06_Glossar/Glossar.docx

Noser, H. (2013). *AVT Arbeitsauswahl*. Von SA-Ausschreibung: avt.hsr.ch abgerufen
Reisacher, D. (2013). Spezifikation (SA Ausgangslage). *Rollen-ManagementTool-Teil II*. IPG AG.

12.4 Dokumentübersicht

Nachfolgend folgt eine allgemeine Beschreibung des Produktes, mit dessen Perspektiven, Funktionen, Einschränkungen und Abhängigkeiten. Danach folgen die spezifischen Anforderungen mit den Qualitätsmerkmalen, wie funktionale bzw. nicht funktionale Anforderungen, der Schnittstellenbeschreibung und den Randbedingungen. Zu guter Letzt, werden die Use Cases beschrieben.

13. Allgemeine Beschreibung

13.1 Produkt Perspektive

Das Rollen Management Tool (RMT) ist eine Lösung um die Rollen zu pflegen. In dieses Tool soll die Verwaltung und Wartung der Rollen, der eigentliche Life Cycle einer Rolle, ermöglichen. RMT soll von einem Rolemining oder einem IAM Tool alle Rollen und Rechte via .CSV-Datei importieren und mittels Export ein IAM Tool mit den Rollen und Rechten versorgen.
Da noch kein ähnliches Produkt, wie RMT auf dem Markt verfügbar ist und es ein Bedürfnis der Kunden von IPG AG abdeckt, soll RMT nach Abschluss der SA, weiterentwickelt und erweitert werden.

13.2 Produkt Funktion

Alle in den Rollen enthaltenen und zu den Rechten gehörenden Informationen können mittels einer .CSV-Datei importiert werden, um RMT mit den benötigten Daten zu beliefern oder dessen Daten zu aktualisieren. Das Rollenmanagement Tool kann auch alle die im System vorhanden Daten in .CSV-Dateien exportieren, damit sie in ein Provisionierungs-Tool geladen werden können.
Man kann mit dem Tool Rollen erstellt und verwaltet werden. Auch können mehrere Rollenversionen in einer Rolle erstellt und bearbeitet werden. Es können mittels älteren Rollen bzw. Rollenversionen neue erstellt werden. In RMT kann der Anwender sich alle Rollen, deren Rollenversionen, sowie die ganzen Rechte nach Applikationen gegliedert anzeigen lassen.

13.3 Benutzer Charakteristik

Kunden (Firmen) von IPG AG

13.4 Einschränkungen

Das Rollen Management Tool ist kein Rolemining Tool und es provisioniert auch keine Berechtigungen. (Reisacher, 2013)

Da es zeitlich nicht in den Umfang der Semesterarbeit passt, muss keine Sicherheit implementiert werden. Es sollen aber die wichtigsten Themen dazu im Punkt 3.1.1.4 aufgelistet werden.

13.5 Abhängigkeiten

RMT hängt von den gelieferten Daten in denen Rechte, Applikationen und Organisationsentitäten (z.B. Organisationseinheiten Kürzel / Name) enthalten sind. Es müssen Rollentypen vorhanden sein. Diese werden für die Gliederung und somit die Übersichtsverbesserung gebraucht. Die Rollentypen können entweder direkt mit der .CSV-Datei der mit dem die Rollen importiert werden dem System hinzugefügt werden oder sie müssen zuerst manuell in RMT erfasst werden. Bevor Rollen importiert oder erstellt werden können, muss das Rollenmanagement Tool mit den Rechten, den Applikationsinformationen und Organisationsentitäten und mindesten einem Rollentyp initialisiert werden.

14. Spezifische Anforderungen (Specific Requirements)

14.1 Qualitätsmerkmale

14.1.1 Funktionale Anforderungen

- Priorität A
  - Eine Rolle muss mit all ihren Versionen abgespeichert werden
  - Diese Rolle und ihre Versionen müssen abgerufen werden können
  - Rollen müssen von einem .csv File eingelesen werden können
  - Rollen müssen in ein .csv File geschrieben werden können, so dass ein IAM System diese einlesen und entsprechende Berechtigungen verteilen kann
  - Lösung für ein Mandant
- Priorität B
  - Es muss ersichtlich sein, wer die Rolle wann und wozu angepasst, sprich eine neue Version erzeugt, hat
  - Die Rollen müssen geringfügig verändert werden können
- Priorität C
  - Die Lösung sollte auf dem Web als Service genutzt werden können
  - 1 bis N Mandanten „optional“

- Die Lösung läuft zuerst im Intranet der IPG AG.
- Später soll die Lösung vom Web aus zugänglich sein.

14.1.1.1 Angemessenheit (Suitability)

Die Sicherheitsdisposition muss noch nicht vorhanden sein. Sollte jedoch in den Grundzügen erwähnt werden, wie diese aussehen könnte.
Von den funktionalen Anforderungen müssen die Funktionalitäten mit Priorität A sicher entwickelt werden. Wenn noch Zeit zu Verfügung steht, sollen danach zuerst diejenigen mit Priorität B und dann mit Priorität C entwickelt werden.

14.1.1.2 Richtigkeit (Accuracy)

Die Lösung soll etwa 400 bis 600 Rollen mit ca. 1000 Rechten pro Rolle, sprich etwa 400‘000 – 600‘000 Rechte verwaltet werden können.
Über die gesamte Applikation sollten in etwa 3-4 Mio. Datenbankobjekte verwaltet werden können

14.1.1.3 Interoperabilität (Interoperability)

RMT muss mit Rolemining- bzw. Provisionierungstools via .CSV-Dateien Daten austauschen können. Von den beiden eben genannten Systemen müssen Rollen und Rechte Mittels .CSV-Datei in RMT importiert werden können.
Nur für die Provisionierungstools müssen die vorhanden Rollen und Rechte im Rollenmanagement Tool als .CSV-Datei exportiert werden können.

14.1.1.4 Sicherheit (Security)

- Die Datenübertragung soll SSL/TLS verschlüsselt sein.
- Es soll ein Login erstellt werden, um den Zugriff nur den berechtigten Personen zu gestatten
- Gängige Attacken sollten bestmöglichst verhindert werden. Z.B. SQL Injection

14.1.1.5 Ordnungsmässigkeit (Functionality Compliance)

Es bestehen keine rechtlichen Bestimmungen oder Vorschriften, welche RMT erfüllen muss.
Beim Schutz der Datenbank bzw. des Applikation Servers und der Verbindungen dazwischen, sowie den Verbindungen zwischen Applikation Server und Client sollten Verschlüsselungen eingesetzt, um die Integrität zu gewährleisten.
Auch soll der Zugriff nur Berechtigten Personen gewährt werden und dies mit einem Authentifizierungsmechanismus, wie z.B. einem Login oder einem Zugriffsschutz zum Netzwerk gewährleistet werden.

14.1.2 Nicht-Funktionale Anforderungen

14.1.2.1 Benutzbarkeit (Usability)

14.1.2.1.1 Verständlichkeit (Understandability)

Das GUI und die Bedienung sollten selbsterklärend und höchstens mit einer kurzen Einführung verwendbar sein.

14.1.2.1.2 Lernbarkeit (Learnability)

Die Handhabung mit dem Rollenmanagement Tool soll einfach zu erlernen sein.

14.1.2.1.3 Bedienbarkeit (Operability)

RMT soll intuitiv bedienbar sein.

14.1.2.1.4 Attraktivität (Attractiveness)

Das GUI muss anschaulich sein.

14.1.2.1.5 Konformität (Usability Compliance)

Es gibt keine Normen und Vereinbarungen die eingehalten werden müssten.

14.1.2.2 Zuverlässigkeit (Reliability)

14.1.2.2.1 Reife (Maturity)

RMT muss am Ende der Arbeit eine lauffähige Version mit mindestens den Kernfunktionen, welche als Priorität A bei Punkt 3.1.1 beschrieben werden, sein.

14.1.2.2.2 Fehlertoleranz (Fault Tolerance)

RMT soll ein geringe bis mittlere Fehlertoleranz aufweisen.

14.1.2.2.3 Wiederherstellbarkeit (Recoverability)

RMT muss wiederherstellbar sein. Was durch den Import bzw. Export der Daten gewährleistet ist.

14.1.2.2.4 Konformität (Reliability Compliance)

Es gibt keine Normen und Vereinbarungen, welche durch RMT eingehalten werden müssen.

14.1.2.3 Effizienz (Efficiency)

14.1.2.3.1 Zeitverhalten (Time Behaviour)

Das Rollenmanagement Tool soll eine akzeptable Performance nach Beendigung der SA aufweisen.

14.1.2.3.2 Verbrauchsverhalten (Resource Utilisation)

Es gibt keine Anforderungen bezüglich dem Verbrauchsverhalten, welche an RMT gestellt werden.

14.1.2.3.3 Konformität (Efficiency Compliance)

Es gibt keine Normen und Vereinbarungen, welche durch RMT eingehalten werden müssen.

14.1.2.4 Wartbarkeit (Maintainability)

14.1.2.4.1 Analysierbarkeit (Analyzability)

Es soll eine Analysierbarkeit nach Software Engineering s Standard in RMT erreicht werden.

14.1.2.4.2 Modifizierbarkeit (Changeability)

Das RMT muss so entwickelt werden, dass es modularisierbar ist, und somit einfach modifizierbar ist, indem neue Module hinzugefügt werden können.

14.1.2.4.3 Stabilität (Stability)

Das Rollenmanagement Tool sollte stabil laufen. Es darf jedoch zu vereinzelten Ausfällen kommen.

14.1.2.4.4 Testbarkeit (Testability)

Der Aufwand zum Testen der Software sollte nicht zu gross sein.

14.1.2.4.5 Konformität (Maintainability)

Es gibt keine Normen und Vereinbarungen, welche durch RMT eingehalten werden müssen.

14.1.2.5 Übertragbarkeit (Portability)

14.1.2.5.1 Anpassbarkeit (Adaptability)

Man soll das Rollenmanagement Tool mit geringem Aufwand an verschiedene Umgebungen anpassen können.

14.1.2.5.2 Installierbarkeit (Installability)

Der Aufwand zum Installieren von RMT in einer festgelegten Umgebung sollte nicht gross sein.

14.1.2.5.3 Koexistenz (Co-Existence)

RMT soll kein Problem haben neben einer Software mit ähnlichen oder gleichen Funktionen zu arbeiten.

14.1.2.5.4 Austauschbarkeit (Replaceability)

Es werden keine spezifischen Forderung an die Austauschbarkeit

14.1.2.5.5 Konformität (Portability Compliance)

Es gibt keine Normen und Vereinbarungen, welche durch RMT eingehalten werden müssen.

14.2 Schnittstellen

Schnittstelle zu Rolmining Tools via .CSV Datei

- Nur Import

Schnittstelle zu IAM Tools via .CSV Datei

- Export
- Import

14.3 Randbedingungen

Maximale Zeilenanzahl für Rollen .CSV Datei: 6‘500

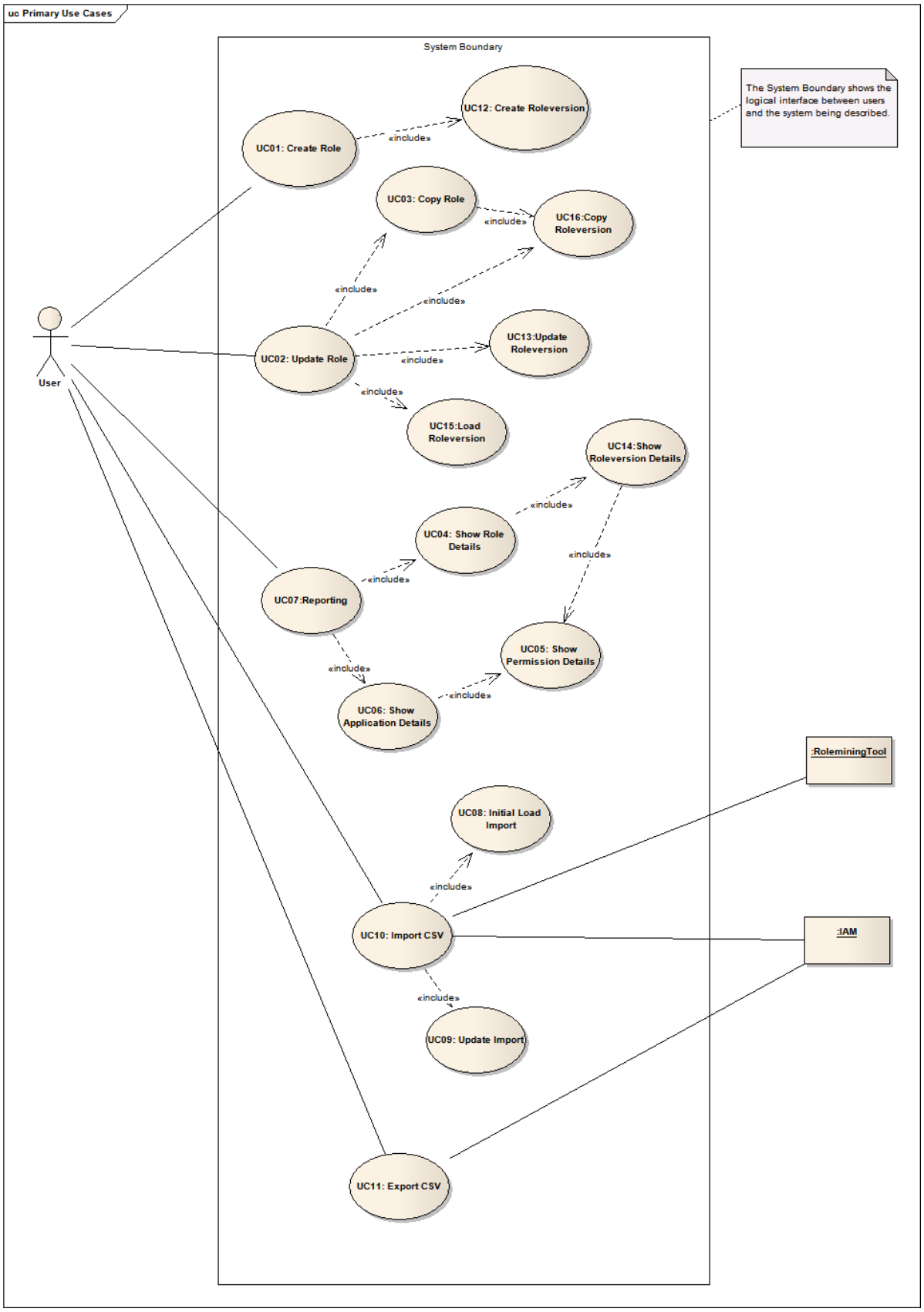
- Aus Performancegründen soll eine Rollen .CSV Datei maximal 6‘500 Zeilen enthalten. Dies kann jedoch je nach Leistung des darunterliegenden Systems variieren und müsste dementsprechend überprüft werden. Die 6‘500 für die Rollen .CSV Datei wurden jedoch getestet mit den Rechnern im SA Raum, welche im Projektplan.docx unter Punkt 7. Infrastruktur beschrieben sind.

Maximale Zeilenanzahl für Rechte .CSV Datei: 25‘000

- Aus Performancegründen soll eine Rechte.CSV Datei maximal 25‘000 Zeilen enthalten. Dies kann jedoch je nach Leistung des darunterliegenden Systems variieren und müsste dementsprechend überprüft werden. Die 25‘000 für die Rechte .CSV Datei wurden jedoch getestet mit den Rechnern im SA Raum, welche im Projektplan.docx unter Punkt 7. Infrastruktur beschrieben sind.

15. Use Cases

15.1 Use Case Diagramm



15.2 Aktoren & Stakeholder

Aktor & Stakeholder	Beschreibung
User	Der User von RMT ist sowohl Aktor, als auch Stakeholder. Er kann Rollen, Rechte und Applikationen (später auch als Daten bezeichnet) von einem Rolemining- oder IAM – Tool via .csv-Datei importieren. Er kann auch mittels einer .csv-Datei die gespeicherten Rollen und Berechtigungen in RMT exportieren. Der User kann selber neue Rollen erfassen, bestehende aktualisieren oder kopieren. Der Benutzer hat auch die Möglichkeit sich die verschiedenen Rollenversionen, sowie das Datenblatt von den Rollen, Rechten und Applikationen anzeigen zu lassen.

15.3 Beschreibungen (Brief)

Nr.	Use Case	Beschreibung
UC 01	Create Role	Der Benutzer kann neue Rollen erfassen. UC01 umfasst UC12.
UC 02	Update Role	Der Benutzer kann bestehende Rollen bearbeiten. UC02 umfasst UC03, UC13, UC15 und UC16.
UC 03	Copy Role	Der Benutzer kann bestehende Rollen kopieren. UC03 umfasst UC16.
UC 04	Show Role Details	Der Benutzer kann sich die relevanten Informationen zur ausgewählten Rolle anzeigen lassen. UC04 umfasst UC14 und UC05.
UC 05	Show Permission Details	Der Benutzer kann sich die relevanten Informationen zur von Rechten anzeigen lassen.
UC 06	Show Application Details	Der Benutzer kann die Details von Applikationen und deren Rechte anzeigen lassen. UC06 umfasst UC05.
UC 07	Reporting	Der Benutzer kann sich die Details von Rollen, Rechten und Applikationen anzeigen lassen. Dieser UC inkludiert UC04, UC05, UC06 und UC14.
UC 08	Initial Load Import	Der Benutzer kann, einen „Initial load“ der Daten machen. Das ist der 1. Import, um RMT mit Daten zu versorgen.
UC 09	Update Import	Der Benutzer kann mit einem Import von einer .csv-Datei die bestehenden Daten aktualisieren oder neue Daten hinzufügen.
UC 10	Import CSV	Der Benutzer kann eine .csv-Datei von einem Rolemining- oder IAM-Tool in RMT importieren. UC10 umfasst UC08 und UC09.
UC 11	Export CSV	Der Benutzer kann die Daten als .csv-Datei exportieren, welches von einem IAM-Tool eingelesen werden kann.
UC 12	Create Roleversion	Es können Rollenversionen erstellt werden.
UC 13	Update Roleversion	Es können bestehende Rollenversionen bearbeitet

		werden.
UC 14	Show Roleversion Details	Rollenversionen können angezeigt werden. UC14 umfasst UC05.
UC 15	Load Roleversion	Eine ältere Rollenversion einer Rolle kann geladen werden.
UC 16	Copy Roleversion	Eine Rollenversion kann kopiert werden.

15.4 Beschreibungen (Fully Dressed)

15.4.1 UC01: Create Role

Primary Actor	Benutzer
Overview	Der Benutzer kann neue Rollen erfassen. UC01 umfasst UC12.
Stakeholder and Interests	Benutzer: Das Erfassen einer neuen Rolle soll einfach und intuitiv sein.
Preconditions	• Organisationsentitäten, Applikationen und Rechte sind bereits in RMT vorhanden. • Es wurde mindestens ein Rollentyp erfasst. • Preconditions von UC12 sind erfüllt.
Postconditions	• Es ist noch keine Rolle mit identischem Namen vorhanden. • Rolle wurde erfolgreich abgespeichert. • Postconditions von UC12 sind erfüllt. • Rollenname, Technischer Rollenname sind eindeutig. • Ein Besitzer, Container, OU und Rollentyp muss ausgewählt sein. • Zuerst wird die Rolle gespeichert und dann die Rollenversion.
Main Success Scenario	1. Benutzer erfasst die Rollendetails 2. Siehe UC12 3. Der Benutzer speichert die Rolle mit der Rollenversion von UC12.
Extensions	2a: Die Rolle bzw. Rollenname ist schon vorhanden a. UC02 wird ausgeführt.
Special Requirements	-
Frequency	Oft

15.4.2 UC02: Update Role

Primary Actor	Benutzer
Overview	Der Benutzer möchte eine bestehende Rolle aktualisieren bzw. anpassen. UC02 umfasst UC03, UC13, UC15 und UC16.
Stakeholder and Interests	Benutzer: Das Bearbeiten soll einfach und intuitiv sein.
Preconditions	• Organisationsentitäten, Applikationen und Rechte sind bereits in RMT vorhanden. • Es wurde mindestens ein Rollentyp erfasst. • Es ist mindestens eine Rolle vorhanden • Die zu bearbeitende Rolle wurde geladen, sofern der Benutzer nicht alle Daten manuell eingeben will in der

	Maske, um neue Rolle zu erstellen. • Je nach Sub UC müssen deren Preconditions erfüllt sein.
Postconditions	• Rollenname, Technischer Rollenname sind eindeutig. • Ein Besitzer, Container, OU und Rollentyp muss ausgewählt sein. • Die bearbeitende Rolle wurde richtig gespeichert • Der Rollenname darf nicht geändert werden sein, sonst siehe UC03 • Je nach Sub UC müssen deren Postconditions erfüllt sein.
Main Success Scenario	1. Benutzer bearbeitet gewünschte Rollenattribute. 2. Siehe UC13 3. Benutzer speichert die Rolle.
Extensions	1a: Der Benutzer gibt alle Rollenattribute manuell in der Maske, um neue Rollen zu erstellen, ein. 1b:Der Benutzer ladet eine Rollenversion und kopiert diese
Special Requirements	-
Frequency	Relativ oft

15.4.3 UC03: Copy Role

Primary Actor	Benutzer
Overview	Der Benutzer kann bestehende Rollen kopieren. UC03 umfasst UC16.
Stakeholder and Interests	Der Benutzer will anhand einer bestehenden Rolle eine neue erstellen.
Preconditions	• Mindestens eine Rolle muss vorhanden sein. • Preconditions von UC16 sind erfüllt
Postconditions	• Eine neue Rolle muss erstellt worden sein. • Rollenname, Technischer Rollenname sind eindeutig. • Ein Besitzer, Container, OU und Rollentyp muss ausgewählt sein. • Postconditions von UC01 und UC13 sind erfüllt
Main Success Scenario	1. Benutzer lädt eine Rolle 2. Benutzer ändert den Rollennamen 3. Benutzer bearbeitet die gewünschten Rollenattribute
Extensions	1a: Der Benutzer gibt alle Rollenattribute manuell in der Maske, um neue Rollen zu erstellen, ein.
Special Requirements	-
Frequency	Relativ oft

15.4.4 UC04: Show Role Details

Primary Actor	Benutzer
Overview	Der Benutzer kann sich die relevanten Informationen zur ausgewählten Rolle anzeigen lassen. UC04 umfasst UC14 und UC05.
Stakeholder and Interests	Benutzer will die Rolle Daten der ausgewählten Rolle einsehen können.
Preconditions	• Es ist mindestens eine Rolle vorhanden

	• Preconditions von UC14 und UC05 sind erfüllt
Postconditions	• Die gewählte Rolle wurde geladen • Die gewählte Rolle wurde korrekt angezeigt. • Postconditions von UC14 und UC05 sind erfüllt.
Main Success Scenario	1. Benutzer wählt eine Rolle aus. 2. Die Details der Rolle werden angezeigt.
Extensions	-
Special Requirements	-
Frequency	Relativ oft

15.4.5 UC05: Show Permission Details

Primary Actor	Benutzer
Overview	Der Benutzer kann sich die relevanten Informationen zur von Rechten anzeigen lassen.
Stakeholder and Interests	Benutzer möchte Details von Rechten einsehen.
Preconditions	• Rechte sind im System vorhanden. • Applikationen sind im System vorhanden.
Postconditions	Rechte werden angezeigt nach Applikationen gegliedert angezeigt.
Main Success Scenario	1. Benutzer wählt den Tab indem die Rechte nach Applikationen gegliedert angezeigt werden. 2. Benutzer wählt Applikation aus zu dem das gewünschte Rechte gehört. 3. Benutzer sucht nach dem Recht. 4. Informationen vom entsprechenden Recht wird angezeigt.
Extensions	1a: Benutzer lädt eine Rolle a. Benutzer lässt sich die Rechte einer Rollenversion anzeigen.
Special Requirements	-
Frequency	Selten-Mittel

15.4.6 UC06: Show Application Details

Primary Actor	Benutzer
Overview	Der Benutzer kann die Details von Applikationen und deren Rechte anzeigen lassen. UC06 umfasst UC05.
Stakeholder and Interests	Benutzer möchte die Details zu einer Applikation übersichtlich angezeigt bekommen.
Preconditions	Preconditions von UC05 sind erfüllt.
Postconditions	Postconditions von UC05 sind erfüllt.
Main Success Scenario	1. Benutzer wählt den Tab indem die Applikation mit ihren Rechten angezeigt werden. 2. Benutzer sucht sich die gewünschte Applikation aus. 3. Die Details zur Applikation werden angezeigt.
Extensions	-
Special Requirements	-
Frequency	Selten - Mittel

15.4.7 UC07: Reporting

Primary Actor	Benutzer
Overview	Der Benutzer kann sich die Details von Rollen, Rollenversionen, Rechten und Applikationen anzeigen lassen. Dieser UC inkludiert UC04, UC05, UC06 und UC14.
Stakeholder and Interests	Benutzer möchte von Details von einem der Oberhalb beschrieben Bereiche anzeigen.
Preconditions	• Preconditions von UC04, UC05, UC06 und UC14 müssen erfüllt sein
Postconditions	• Postconditions von UC04, UC05, UC06 und UC14 müssen erfüllt sein
Main Success Scenario	1. Benutzer wählt sich den entsprechenden Bereich aus von dem er Details angezeigt haben möchte. 2. Für Rollen und Rolenversionen siehe UC04, UC05 und UC14 3. Für Applikationen siehe UC06 und UC5
Extensions	-
Special Requirements	-
Frequency	Oft

15.4.8 UC08: Initial Load Import

Primary Actor	Benutzer
Overview	Der Benutzer kann, einen „Initial load“ der Daten machen. Das ist der 1. Import, um RMT mit Daten zu versorgen.
Stakeholder and Interests	Benutzer möchte die Initialen Daten in RMT einspeisen. Es muss einfach sein zu handhaben und eine angemessene Performance aufweisen.
Preconditions	Es sind keine Daten in RMT vorhanden.
Postconditions	• Es wurden zuerst die Rechte eingelesen und erst danach die Rollen. • Die gelieferten Daten wurden korrekt in RMT importiert.
Main Success Scenario	1. Der Benutzer geht zum Tab für den Import / Export. 2. Der Benutzer lässt zuerst die .CSV-Datei für die Rechte einlesen 3. Der Benutzer konfiguriert den Importprozess. 4. Der Benutzer startet den Importprozess. 5. Der Benutzer lässt die .CSV-Datei für die Rollen einlesen. 6. Der Benutzer konfiguriert den Importprozess. 7. Der Benutzer startet den Importprozess.
Extensions	4a: Der Benutzer importiert nur die Rechte.
Special Requirements	• .CSV Datei für Rollen hat maximal 6500 Zeilen. • .CSV Datei für Rechte hat maximal 25‘000 Zeilen.
Frequency	Selten-Mittel

15.4.9 UC09: Update Import

Primary Actor	Benutzer
Overview	Der Benutzer kann mit einem Import von einer .csv-Datei die bestehenden Daten aktualisieren oder neue Daten hinzufügen.

Stakeholder and Interests	Benutzer möchte die vorhandenen Daten in RMT mit einem Import von Rollen oder Rechten aktualisieren. Es muss einfach sein zu handhaben und eine angemessene Performance aufweisen.
Preconditions	• Wenn nur Rollen importiert werden müssen die Rechte schon vorhanden sein.
Postconditions	• Es wurden zuerst die Rechte eingelesen und erst danach die Rollen, ausser die Daten, welche von den Rollen benötigt werden sind in RMT schon vorhanden. • Die gelieferten Daten wurden korrekt in RMT importiert.
Main Success Scenario	1. Der Benutzer geht zum Tab für den Import / Export. 2. Der Benutzer lässt zuerst die .CSV-Datei für die Rechte einlesen 3. Der Benutzer konfiguriert den Importprozess. 4. Der Benutzer startet den Importprozess.
Extensions	1a: Benötigte Daten der Rechte für die Rollen sind schon vorhanden a. Der Benutzer lässt die .CSV-Datei für die Rollen einlesen. b. Der Benutzer konfiguriert den Importprozess. c. Der Benutzer startet den Importprozess. 1b: Benötigte Daten der Rechte für die Rollen sind nicht oder nur teilweise vorhanden a. Zuerst, wie im Main Success Scenario vorgehen b. Dann, wie bei 1a vorgehen
Special Requirements	Siehe Special Requirements UC08
Frequency	Selten-Mittel

15.4.10 UC10: Import CSV

Primary Actor	Benutzer
Overview	Der Benutzer kann eine .csv-Datei von einem Rolemining- oder IAM-Tool in RMT importieren. UC10 umfasst UC08 und UC09.
Stakeholder and Interests	Der Benutzer möchte Rechte und Rollen mittels je einer .CSV-Datei in RMT importieren.
Preconditions	Siehe UC08 und UC09
Postconditions	Siehe UC08 und UC09
Main Success Scenario	Siehe UC08 und UC09
Extensions	Siehe UC08 und UC09
Special Requirements	Siehe Special Requirements UC08
Frequency	Selten-Mittel

15.4.11 UC11: Export CSV

Primary Actor	Benutzer
Overview	Der Benutzer kann die Daten von RMT als .csv-Datei exportieren, welches von einem IAM-Tool eingelesen werden kann.
Stakeholder and Interests	Benutzer möchte die vorhandenen Daten von RMT exportieren. Es sollte einfach zu handhaben sein und eine angemessene Performance haben.
Preconditions	-

Postconditions	• Die Daten für die Rechte wurden korrekt in eine .CSV-Datei korrekt exportiert. • Die Daten für die Rollen wurden korrekt in eine .CSV-Datei exportiert. Gilt nur für die Extension 2a:
Main Success Scenario	1. Der Benutzer geht zum Tab für den Import / Export. 2. Der Benutzer konfiguriert den Exportprozess für den Export der Rechte. 3. Der Benutzer startet den Exportprozess. 4. Der Benutzer wählt den Ort aus, wo die .CSV-Datei gespeichert werden soll.
Extensions	2a: Der Benutzer konfiguriert den Exportprozess für den Export der Rollen
Special Requirements	-
Frequency	Selten-Mittel

15.4.12 UC12 : Create Roleversion

Primary Actor	Benutzer, UC01
Overview	Es können Rollenversionen erstellt werden.
Stakeholder and Interests	• Der Benutzer möchte die Daten für die Rollenversion erfassen. • UC12 muss erfolgreich erfüllt werden bevor UC01 erfolgreich abgeschlossen werden kann.
Preconditions	-
Postconditions	• Das “Gültig von” Datum ist vor “Gültig bis”. • Es muss im Minimum ein “Gültig von” Datum gesetzt werden und ein Ersteller erfasst sein. • Die Rollenversion kann erfolgreich gespeichert werden
Main Success Scenario	1. Der Benutzer erfasst die rollenversion-spezifischen Daten bei der Maske von UC01.
Extensions	-
Special Requirements	-
Frequency	Oft

15.4.13 UC13: Update Roleversion

Primary Actor	Benutzer, UC02
Overview	Es können bestehende Rollenversionen bearbeitet werden.
Stakeholder and Interests	• Benutzer möchte eine bestehende Rollenversion einr bestehenden Rolle bearbeiten. • UC13 muss erfolgreich erfüllt werden bevor UC01 erfolgreich abgeschlossen werden kann.
Preconditions	• Die bestehenden Rollenversionsdaten wurden erfolgreich geladen und zum Bearbeiten angezeigt. • Wenn nur “Gültig von” gesetzt ist, muss zuerst ein “Gültig bis” Datum gesetzt warden bevor die Rollenversion bearbeitet warden kann.
Postconditions	• Wie Postconditions von UC12
Main Success Scenario	1a: Wenn Rolle eine aktive Rollenversion hat und der Rollenname ist geändert worden

	a. UC01 wird ausgeführt 1b: Wenn Rolle eine aktive Rollenversion hat und der Rollenname ist nicht geändert worden Wenn ein „Gültig bis“ Datum gesetzt wurde und das wird eine neue Rollenversion
Extensions	
Special Requirements	
Frequency	Mittel - Oft

15.4.14 UC14 : Show Roleversion Details

Primary Actor	Benutzer
Overview	Rollenversionen können angezeigt werden. UC14 umfasst UC05.
Stakeholder and Interests	Der Benutzer möchte alle Rollenversionen einer Rolle einsehen.
Preconditions	• Die Rollenverionen wurden erfolgreich geladen. • Preconditions von UC05
Postconditions	-
Main Success Scenario	1. Die Details der Rollenversionen der ausgewählten Rolle werden angezeigt.
Extensions	-
Special Requirements	-
Frequency	Oft

15.4.15 UC15 : Load Roleversion

Primary Actor	Benutzer
Overview	Eine ältere Rollenversion einer Rolle kann geladen werden.
Stakeholder and Interests	Der Benutzer möchte eine ältere Rollenversion der ausgewählten Rolle laden.
Preconditions	Es wurde eine Rolle mit ihren Rollenversionen geladen.
Postconditions	Die vom Benutzer gewählte Rollenversion wurde erfolgreich geladen.
Main Success Scenario	1. Der Benutzer wählt die zu ladende Rolle aus und lässt diese in die Bearbeitungsmaske laden.
Extensions	-
Special Requirements	-
Frequency	Oft

15.4.16 UC16 : Copy Roleversion

Primary Actor	Benutzer, UC02, UC03
Overview	Eine Rollenversion kann kopiert werden.
Stakeholder and Interests	Der Benutzer möchte von einer zu kopierenden Rolle die Rollenversionen kopieren. UC02 und UC03 erwarten ein erfolgreiches Abschliessen dieses UC.
Preconditions	Die neueste Rollenversion ist aktiv und hat ein „Gültig bis“ Datum in der Zukunft gesetzt.
Postconditions	• Die Rollenversion wurde erfolgreich kopiert und der

	Rolle angefügt. • Im Minimum muss ein „Gültig von“ Datum gesetzt werden.
Main Success Scenario	1. Der Benutzer ändert die gewünschten Rollenversion Daten 2. Der Benutzer setzt nicht überlappende von/bis Daten.
Extensions	2a: gilt nur, von UC02 gebraucht, ansonsten, wenn von UC03 gebraucht, können alle gewünschten Rollenversionsdaten angegeben werden.
Special Requirements	-
Frequency	Oft

Domainanalyse

Änderungsgeschichte

Datum	Version	Änderung	Autor
21.03.2013	1.0	Initialversion	msondere
30.05.2013	1.1	Aufbereitung & Vervollständigung	fmahler
31.05.2013	1.2	Letzte Verbesserungen	fmahler

Inhalt

16. Einführung

16.1 Zweck

Dieses Dokument beschreibt die Domainanalyse von RMT.

16.2 Gültigkeitsbereich

Dieses Dokument ist gültig über die ganze Dauer der SA und ist für RMT bestimmt. Es werden noch Änderungen, als auch Erweiterungen an diesem Dokument vorgenommen.

16.3 Referenzen

2001 – Craig Larman - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process - ISBN 0-13-092569-1

16.3.1 Definitionen und Abkürzungen

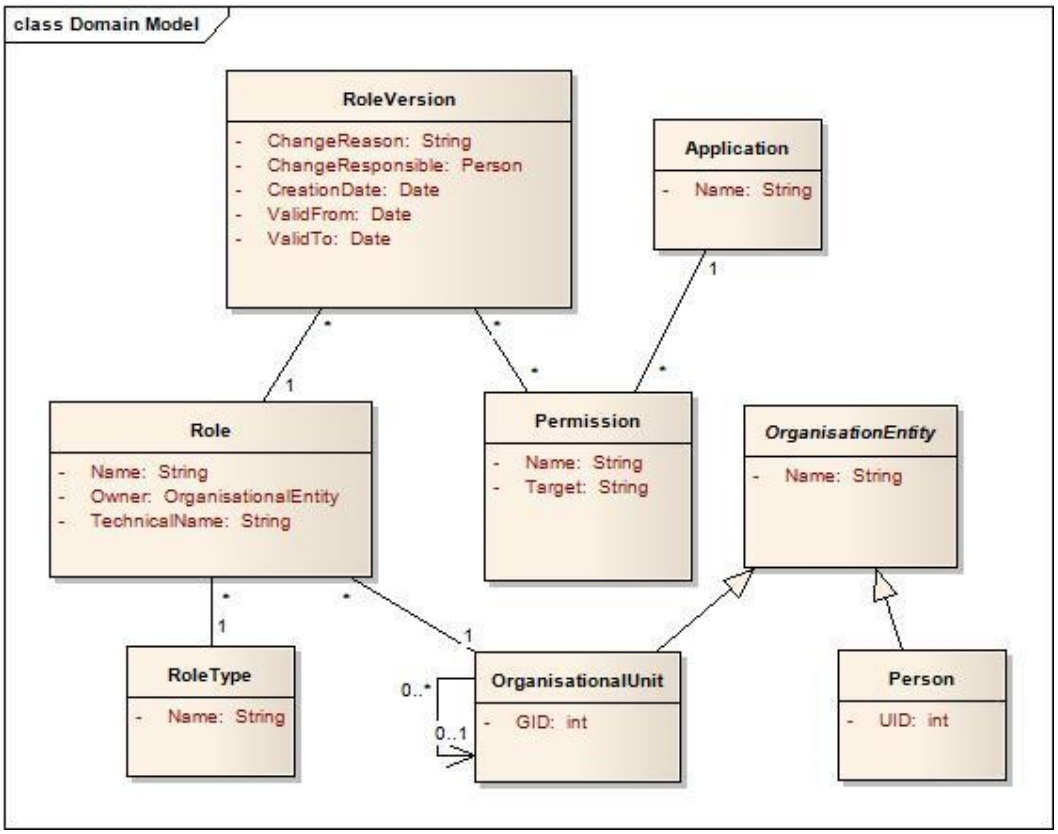
- siehe globales Glossar

16.4 Übersicht

Dieses Dokument beginnt mit der Präsentierung des Domainmodells von RMT und dessen Beschreibung. Anschliessend werden die wichtigsten Konzepte des Domainmodells detailliert erläutert und die wichtigsten Systemsequenzdiagramme gezeigt.

17. Domain Modell

17.1 Strukturdiagramm

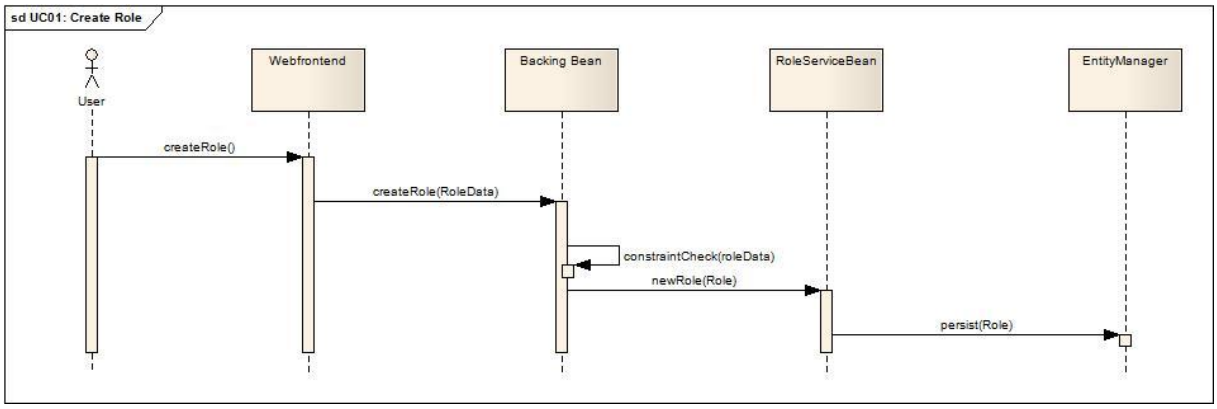


17.2 Wichtige Konzepte

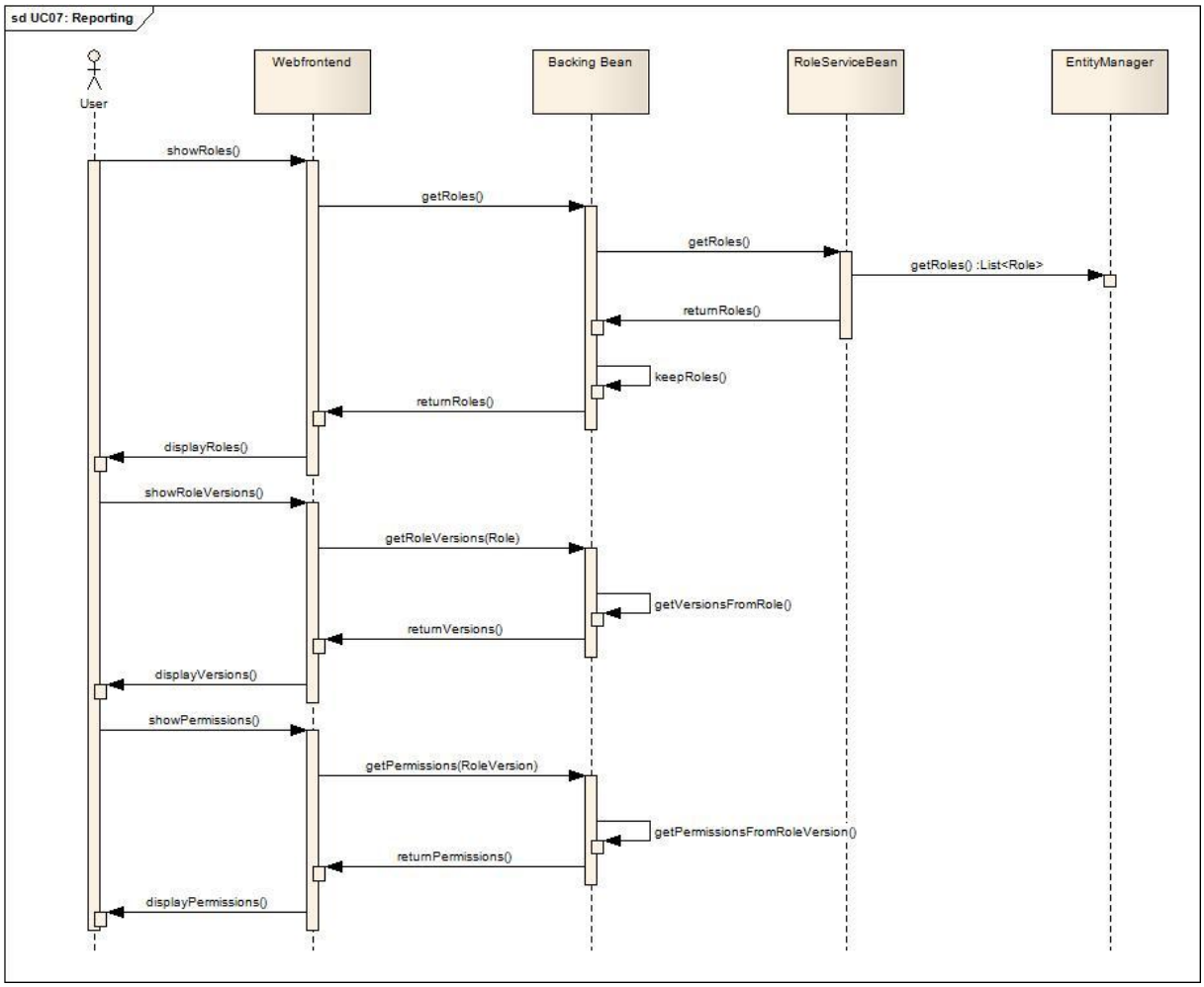
Im Zentrum der Problem Domain steht die Beziehung zwischen Rollen, Rollenversionen und Berechtigungen. Da wir die Versionen einer Rolle auf unbestimmte Zeit historisieren müssen und diese jederzeit abrufbar sein sollen, haben wir uns dazu entschieden, die Versionen als eigene Domänenobjekte zu betrachten. Entsprechend sind in der Klasse „Role“ keinerlei Permissions verlinkt, denn diese dient nur zur Haltung von Metadaten und als Container für ihre Versionen.
Da der Owner einer Rolle sowohl eine Person als auch eine Organisationseinheit sein kann, haben wir die gemeinsame Oberklasse OrganisationEntity eingeführt. Dabei ist zu beachten, dass die OUs vom Zielsystem ausgelesen werden, während die Personen ausschliesslich in unserem System vorhanden sind.

18. Systemsequenzdiagramme

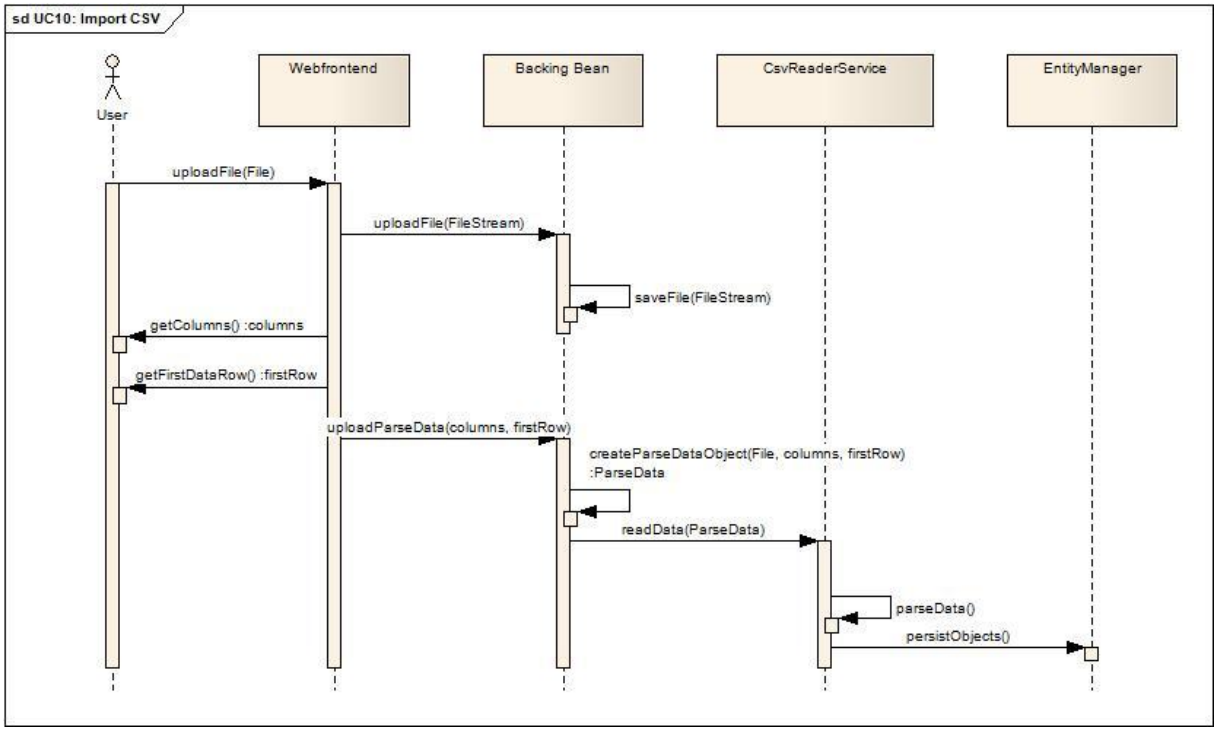
18.1 UC01: Create Role



18.2 UC07: Reporting



18.3 UC10: Import CSV



Software Architektur Dokument

Änderungsgeschichte

Datum	Version	Änderung	Autor
29.05.2013	1	Initiierung	Florian Mahler
31.05.2013	1.1	Aufbereitung & Letzte Verbesserungen	Florian Mahler

Inhalt

19. Einführung

19.1 Zweck

Zweck dieses Dokumentes ist die Architektur der Studienarbeit Rollenmanagement Tool zu erläutern.

19.2 Gültigkeitsbereich

Dieses Dokument bezieht sich ausschliesslich auf die Studienarbeit „Rollenmanagement Tool“.

19.3 Referenzen

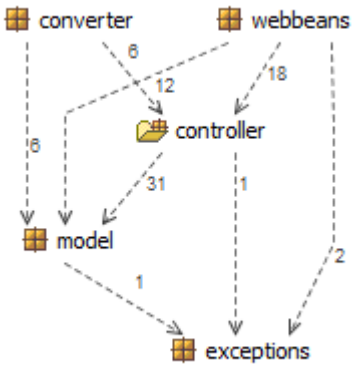
http://docs.oracle.com/javaee/6/api/overview-summary.html

19.4 Übersicht

Dieses Dokument soll eine Übersicht der Architektur von Rollenmanagement Tool aufzeigen, sowie auf architektur- und deploymentrelevante Details eingehen. Sowohl die Laufzeitumgebung als auch die Datenspeicherung werden beschrieben.

20. Systemübersicht

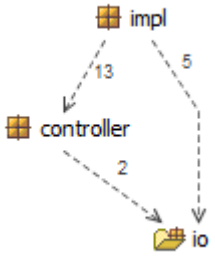
Für dieses Projekt wurde die vorgesehene Standard-Architektur für J2EE-Webapplikationen gewählt. Dadurch wurde die Software in übersichtlicher Weise in einer Model-View-Controller-Architektur entworfen.
Wie der nebenstehenden Grafik zu entnehmen ist, wurde diese grundsätzlich umgesetzt, wobei einige Dinge auffallen:



- Wie es JSF vorsieht, ist im Package „webbeans“ nicht die eigentliche View enthalten, sondern deren „backing Beans“. Diese stellen die Daten für die eigentliche View, die Webseite, bereit, welche sie wiederum von controllern bezieht.
- Das Package „converter“ enthält Konverterklassen, welche Daten, die vor der Darstellung aufbereitet werden müssen, entsprechend konvertiert. Man könnte sie als Teil der View behandeln, allerdings sind wir der Ansicht, dass diese Trennung weitere Übersicht schafft.
- Die softwarespezifischen Fehlermeldungen müssen in allen Packages bekannt sein. Man könnte sie als Teil des Datenmodells betrachten, jedoch entschieden wir uns für ein separates Package „exceptions“, da diese keinen Einfluss auf das eigentliche Datenmodell haben.

Wie im Model-View-Controller-Pattern vorgesehen, beinhaltet das „model“ Package alle Datenmodellklassen. Diese werden mittels JPA persistiert, dabei wurden ausschliesslich Annotations auf Felder und Klassen verwendet. Die setter-Methoden und Konstruktoren stellen sicher, dass die Constraints, welche in den Parametern dieser Annotations vorgegeben sind, auch eingehalten werden, soweit dies möglich ist. Einige Constraints können hingegen nur von ausserhalb überprüft werden, was beim Einfügen von Daten beachtet werden muss. Ebenfalls auf den Modelklassen sind per Annotations die von den Controllern verwendeten Named Queries definiert.

Das „controller“ Package beinhaltet alle Interfaces für die EJBs. Deren Implementationen, das umfasst alle Controllerklassen, sind im Subpackage „impl“ abgelegt, während sich im „io“ Package die Interfaces der Datenobjekte für Parser/Generator befinden. Wir empfehlen, für etwaige andere Parse- und Generationstechniken jeweils ein Subpackage innerhalb des io Package für die Implementationen dieser Interfaces zu verwenden, wie wir es mit dem „csv“ Subpackage bereits gemacht haben. So wird die grosse Flexibilität dieser Technik durch Übersichtlichkeit noch unterstützt.



Controller-Packages

Im webbeans Package befinden sich, wie bereits erwähnt, sämtliche backing Beans. Diese eigentlichen Datenhalter für die jeweiligen Viewkomponenten sind direkt and diese gekoppelt, so wie es der JSF Standard vorsieht. Da, wo die Möglichkeiten dieser Beans nach dem Aspekt von Separation of concerns für gewisse Funktionalitäten nicht ausreichen, aber die Kompetenzen des Controllers bei einer Auslagerung zu ihnen überschritten würden, werden Konverter zur Hilfe gezogen, welche sich alle direkt im converter Package befinden.

21. Architektonische Ziele & Einschränkungen

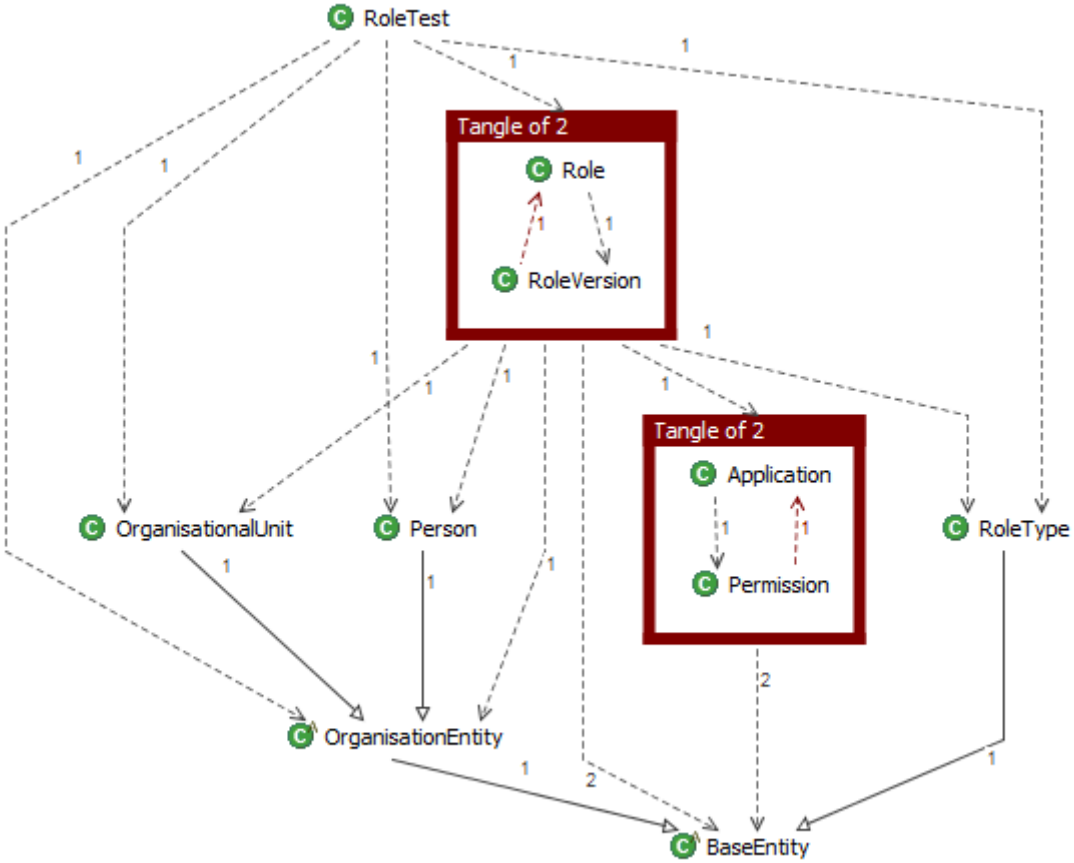
Mit der Verwendung von JPA und EntityManager zur Persistierung der Daten haben wir uns für eine auf Datenkonsistenz ausgerichtete Architektur entschieden. Dies ist aufgrund der ziemlich restriktiven Constraints auf den Datenobjekten und –feldern sicher sinnvoll, da so selbst bei einem Absturz kaum je korrupte Datensätze vorhanden sein werden, kann aber zu Performanceeinbussen führen, wenn viele verschiedene Daten in einer Session geladen werden müssen.
Da wir das Requirement „Login“ aufgrund Zeitmangels zurückgestellt haben, ist Security zunächst ein kleineres Thema. Um hier den Bedürftnissen der Kunden der IPG AG zu entsprechen, müsste man ein Authentifikation- und Autorisierungssystem einführen. Einige damit verbundene Funktionen sind bereits implementiert, werden allerdings nicht verwendet. Alternativ kann man aber auch ein solches System über den Application Server betreiben, auf welchem die Webapplikation läuft. Andere sicherheitsrelevante Punkte, wie die Verhinderung von SQL Injecton Attacken durch Verwendung von vorkompilierten Named Queries, wurden bereits umgesetzt, nützen allerdings ohne ein Authentifikationssystem wenig. Für Übertragungssicherheit wiederum kann nur der Application Server zuständig sein, da unsere Software beispielsweise nicht die Verwendung von SSL/TLS forcieren kann.
Die Software ist monolithisch aufgebaut, sämtliche Logik befindet sich auf einem einzelnen Server. Die Datenbank kann sich bei entsprechender Konfiguration auch an anderer Stelle befinden. Injection von EJBs auf Remotesysteme ist zwar grundsätzlich möglich, hier aber in unseren Augen nicht sinnvoll, so oder so aber vom verwendeten Application Server abhängig. Unter JBoss 7.1.1, welchen wir verwenden, wäre dies beispielsweise nicht möglich.
Besonderes Augenmerk wurde auch auf Erweiterbarkeit gelegt. Durch die J2EE Architektur wird es extrem einfach, Views hinzuzufügen und mit backing Beans zu versehen, neue EJBs als Controller einzurichten oder gar Schemaerweiterungen vorzunehmen. Es werden nur neue Abhängigkeiten geschaffen, wenn Erweiterungen am Datenmodell vorgenommen werden – und dann sind neue Abhängigkeiten in jedem Fall unvermeidbar – oder wenn sie bewusst geschaffen werden, um gewisse neue Funktionalitäten hinzuzufügen. Wichtig ist, dass neue Model-Klassen von der abstrakten Klasse BaseEntity erben, da diese wesentliche Elemente für den stabilen Betrieb und die Wartbarkeit beinhaltet.

22. Logische Architektur

<Beschreibung der logischen Struktur des Projekts. Pro Subsystem/Package ein einzelner Abschnitt und ein Übersichtsdiagramm über die einzelnen Subsysteme/Packages. Aufteilung in Subsysteme/Packages (zum Beispiel: 3-Layer-Architektur mit GUI, Problem Domain und Datenhaltung). >

22.1 Model

22.1.1 Klassenstruktur



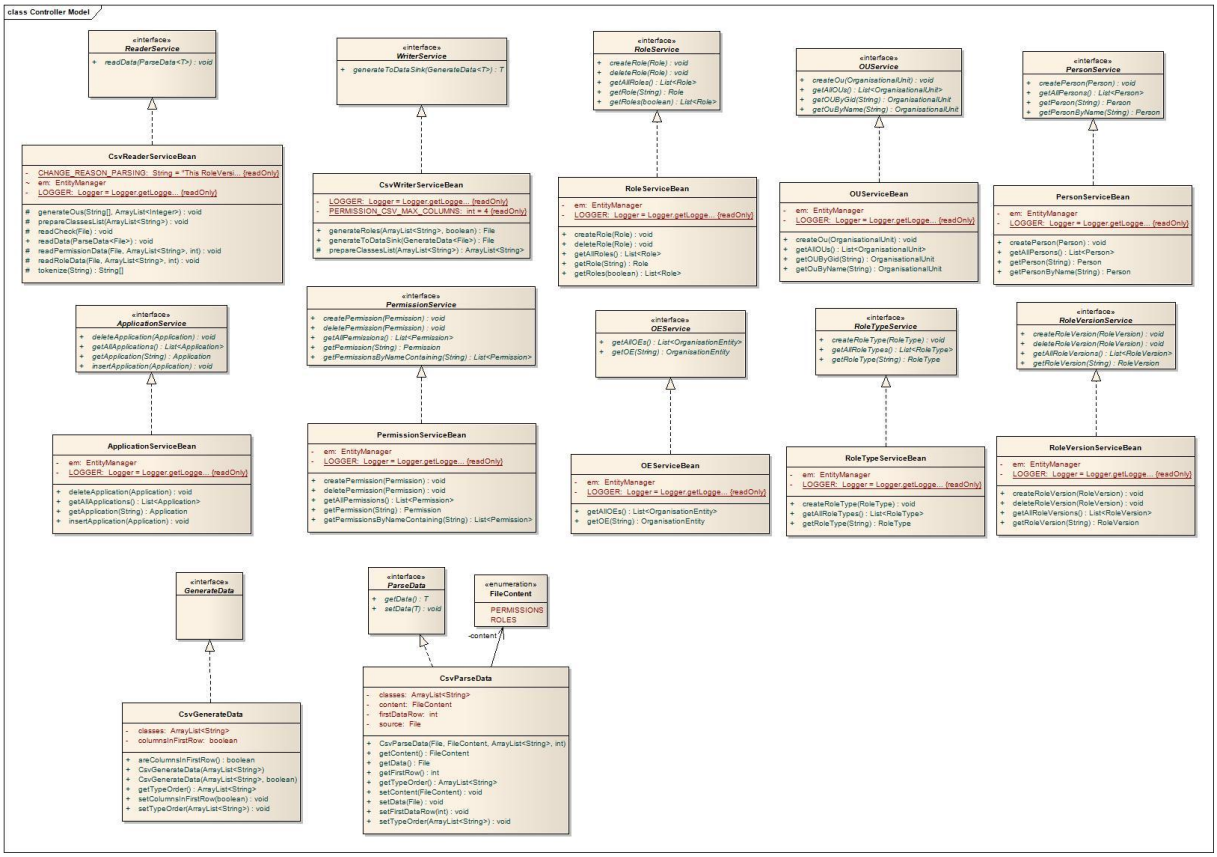
Die beiden Klassenpaare Application-Permission und Role-Roleversion haben jeweils unschöne gegenseitige Beziehungen. Dies wurde so gestaltet, um die Datenabfrage zu beschleunigen. Bei beiden Paarungen besteht die Gefahr, dass diese bei wachsenden Datenmengen stark beeinträchtigt wird; bei den Rollenversionen durch die konstant wachsende Historie, bei den Permissions durch ihre schiere Anzahl.
Die abstrakte Klasse BaseEntity wurde eingeführt, um verschiedene Aufgaben zu übernehmen, welche in allen Klassen erledigt werden müssen. So speichert sie einen Namen, generiert bei jeder Persistierung einen Zeitstempel und stellt einen von sämtlichen Daten unabhängigen Primärschlüssel bereit, welcher mit der gewählten Strategie auf jeder möglichen Datenbank generiert werden kann. Für die Verwendung von Optimistic Locks ist auch ein Versionsfeld vorhanden.
Die Objekte der Klassen „Person“ und „OrganisationalUnit“ werden in einer gemeinsamen Tabelle „OrganisationEntity“ abgelegt, wobei der Delimiter für Personen „P“, derjenige für Organisationseinheiten „OU“ ist.

22.1.2 Schnittstellen

Alle zu persistierenden Datenstrukturen werden nur über die Interfaces List<T>, Set<T> und Map<K,V> definiert. Die Laufzeitimplementation dieser Datenstrukturen wird von der JPA definiert, Zugriffe finden nur über die Methoden dieser Interfaces statt.

22.2 Controller

22.2.1 Klassenstruktur



Controller Classes

Im Controller-Package zeigt sich die schlichte Eleganz der J2EE Architektur. Sämtliche Controllerklassen sind als EJBs implementiert, werden also vom Application Container instanziert und per Dependency Injection in denjenigen Teilen der View beigezogen, welche sie brauchen. Jedes EJB implementiert ein spezifisches Interface, welches für die Injection verwendet wird.
Im Subpackage „io“ befinden sich Interfaces für die Datenobjekte des Parsers und des Generators, so dass generisch definiert werden kann, wie ein solcher Parser und Generator aussehen kann, und mit minimalem Aufwand und grösstmöglicher Flexibilität neue Techniken nach Bedarf eingeführt werden können. Implementiert wurden nach Vorgabe ausschliesslich je ein Parser und Generator für .csv Dateien. Die dazu gehörenden Datenobjekte befinden sich im Subpackage controller.io.csv.

22.2.2 Schnittstellen

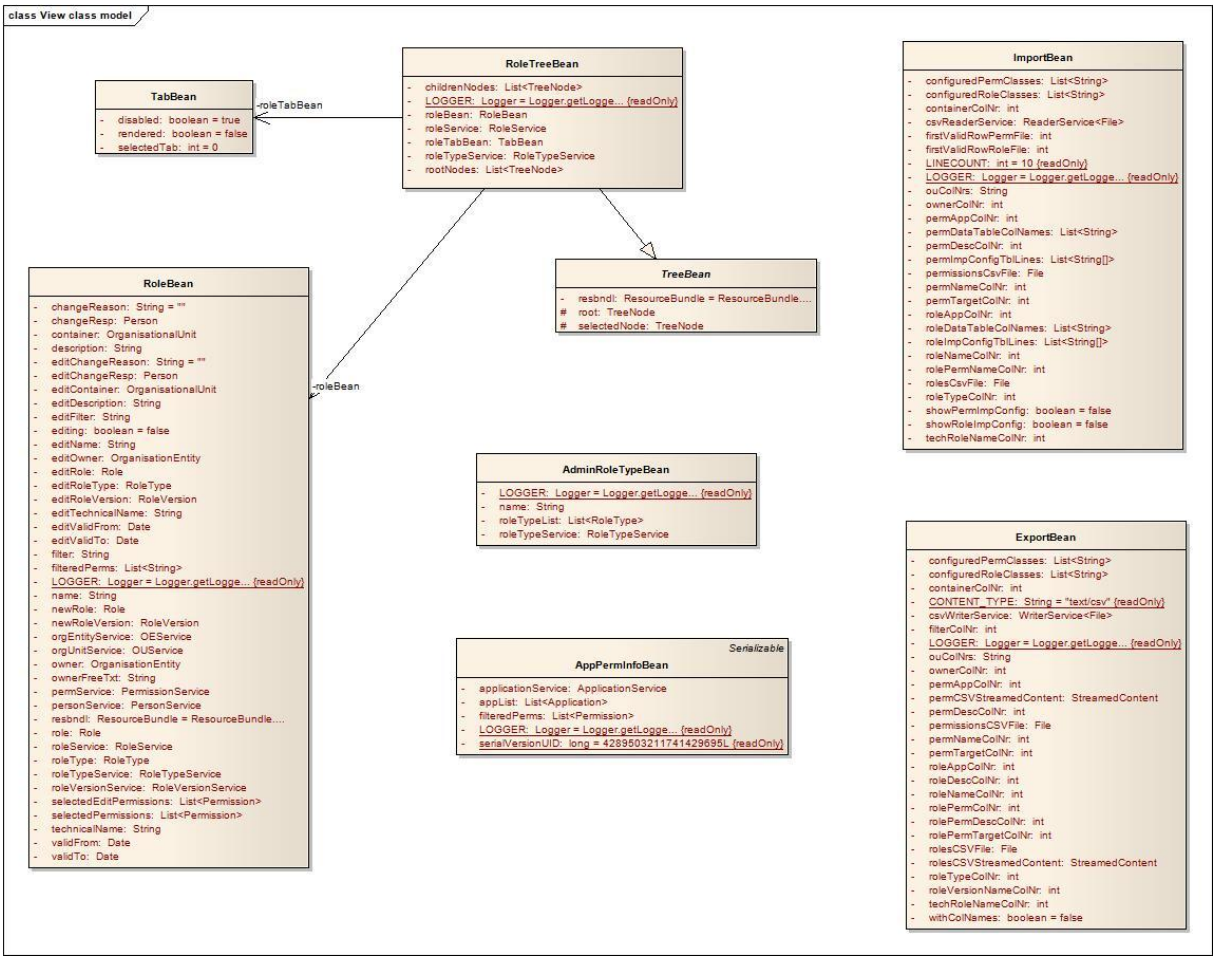
Für jeden per Dependency Injection einzusetzenden Controller existiert ein Interface, welches von diesem Controller implementiert werden kann, ebenso für die Datenobjekte der ReaderService/WriterService – Klassen.

22.2.3 Wichtige interne Abläufe

Die Controller werden ausschliesslich mittels Dependency Injection eingesetzt. Da diese über die jeweiligen Interfaces stattfindet, müssen alle öffentlichen Methoden der Controller in den Interfaces definiert sein, ansonsten können sie nicht verwendet werden.

22.3 Webbeans

22.3.1 Klassenstruktur



Wenn gewisse Klassen in diesem Diagramm etwas überladen aussehen, dann liegt dies daran, dass sie die gesamten Daten der View halten müssen.

22.3.2 Interfaces

In diesem Package sind keine Interfaces vorhanden. Es werden jedoch diejenigen der Controller für die Dependency Injection verwendet.

23. Deployment

23.1 Application Server

Der Application Server ist das Herzstück der Laufzeitumgebung. Er stellt den Application Container für die J2EE Objekte zur Verfügung, instantiert die EJBs, verbindet die backing Beans mit der Webseite und stellt der JPA die Schnittstelle zur Datenbank zur Verfügung. Sämtliche J2EE APIs sind nur hier verfügbar.
Für die Entwicklung haben wir als Application Server JBoss 7.1.1 für Windows verwendet, welcher auch der Abgabe beiliegt. Es kann auch ein beliebiger anderer AS verwendet werden, jedoch muss sichergestellt sein, dass folgende Module (unter Beibehaltung der Entwicklungskonfiguration) geladen werden:

- Javaee-api 6.0
- JRE 1.7.0_17
- Eclipselink 2.4.1
- Jsf-api 2.1.2
- Jsf-impl 2.1.2
- Primefaces 3.5
- Mysql-connector-java 5.1.24
- Log4j 1.2.17

Diese Liste ist solange gültig, wie die Konfiguration verwendet wird, welche wir für die Entwicklung verwendet haben.
Aus Gründen der Einfachheit wurden in unserem Projekt die DB-Zugriffsdaten im persistence.xml abgelegt, was in einer produktiven Umgebung anders gemacht sein sollte. Wir empfehlen, die DB-Zugriffsdaten in der Konfiguration des AS zu konfigurieren.

23.2 Java VM

Die Java Runtime Environment ist verantwortlich für die Ausführung von kompilierten Java Programmteilen. Im Regelfall wird diese gleich innerhalb des AS bereitgestellt, ansonsten muss sichergestellt sein, dass eine J2SE Runtime mindestens in der Version 1.7 Build 17 vorhanden ist.

23.3 Webserver

Der Webserver wird im Regelfall durch den AS bereitgestellt. Sollte dies bei einer anderen Konstellation nicht der Fall sein, ist zu überprüfen, ob und wie die Daten zwischen AS und Webserver ausgetauscht werden (können).

23.4 Datenbank

Grundsätzlich kann eine beliebige Datenbank verwendet werden, für welche Treiber und AS-Libraries zur Verfügung stehen. Im persistence.xml wurde für die Entwicklung konfiguriert, dass ein MySQL Server 5.6.10 verwendet wird. Sollte eine andere DB zum Zuge kommen, muss diese Konfiguration angepasst werden, ebenso falls die DB auf einem Remote-Rechner statt auf demselben Server läuft. Die anzupassenden Einträge im persistence.xml sind:

```
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
<property
name="javax.persistence.jdbc.url"value="jdbc:mysql://localhost:3306/rmtdb"/>
<property name="eclipselink.target-database" value="MySQL"/>
```

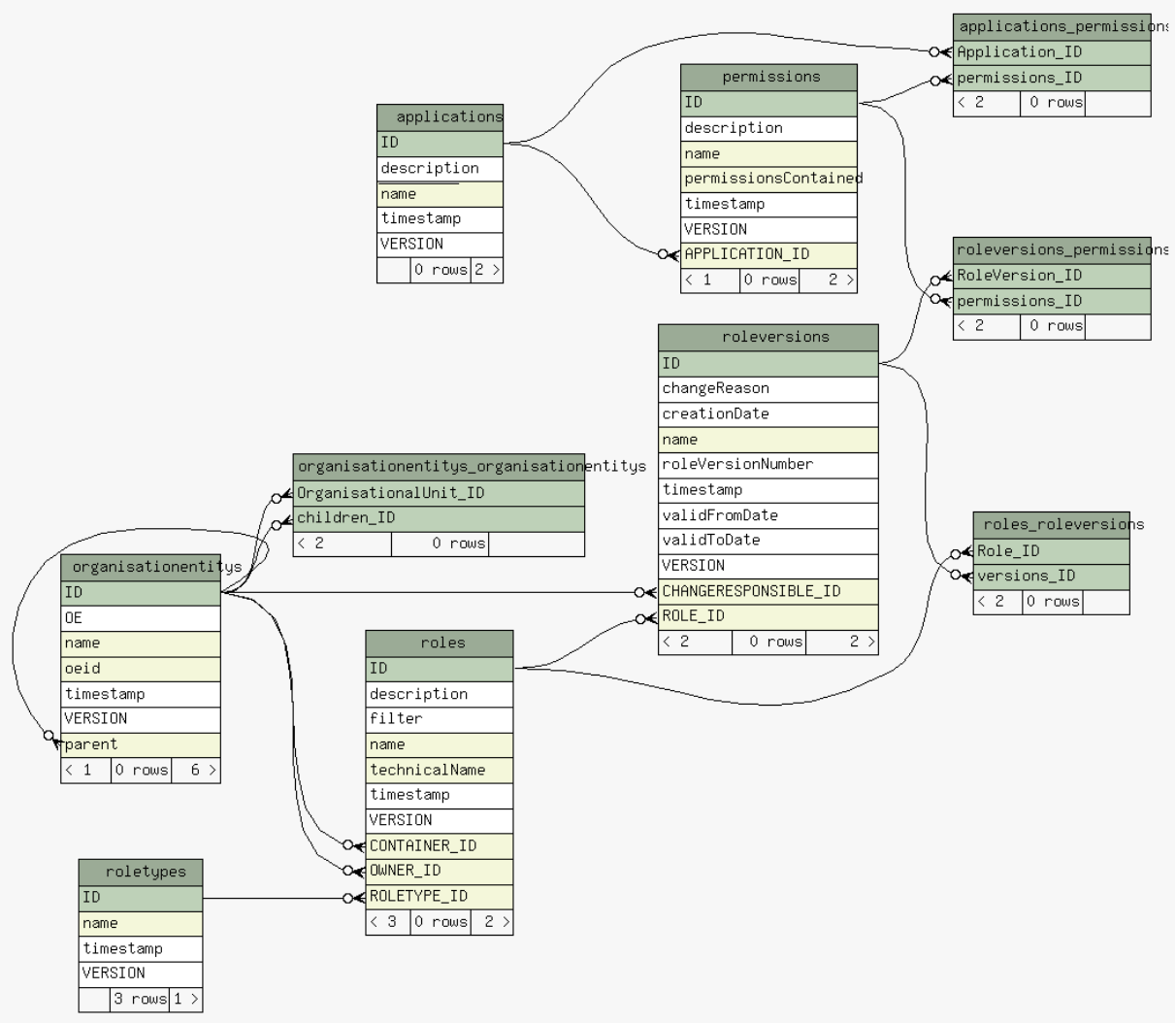
Die passenden Konfigurationen im persistence.xml für die gewünschte Datenbank sei der Dokumentation des jeweiligen Herstellers zu entnehmen.
Falls es erwünscht sein sollte, eine andere Implementation der JPA 2.0 zu verwenden als, wie in unserem Falle, EclipseLink, dann müssen alle eclipselink-spezifischen Properties ausgetauscht und entsprechende Libraries zur Verfügung gestellt werden. An der Implementation der Software muss hingegen nichts verändert werden.

23.5 Maven

Für die einfache, eindeutige und portable Importdefinition der externen Libraries haben wir Maven 3.1 verwendet. Auch der abgegebene Build unserer Software wurde mit Maven erstellt. Dies bietet viele Vorteile, zum Beispiel das einfache Hinzufügen von Sourcecode und Javadoc zu den Libraries oder das einfache Wechseln auf neue Versionen von verwendeten externen Komponenten, aber auch potentielle Fehlerquellen. Sollten auf einem anderen AS Probleme auftreten, empfiehlt es sich, den deployment scope der verantwortlichen Libraries im pom.xml mit den im AS vorhandenen Modulen abzugleichen und einen neuen Build zu erstellen.

24. Datenspeicherung

24.1 ERM



24.2 Table Constraints

Constraint Name	Child Column	Parent Column	Delete Rule
FK_Applications_Permissions_Application_ID	applications_permissions.Application_ID	applications.ID	Restrict delete
FK_Applications_Permissions_permissions_ID	applications_permissions.permissions_ID	permissions.ID	Restrict delete
FK_OrganisationEntitys_parent	organisationentitys.parent	organisationentitys.ID	Restrict delete
FK_Permissions_APPLICATION_ID	permissions.APPLICATION_ID	applications.ID	Restrict delete
FK_Roles_CONTAINER_ID	roles.CONTAINER_ID	organisationentitys.ID	Restrict delete
FK_Roles_OWNER_ID	roles.OWNER_ID	organisationentitys.ID	Restrict delete
FK_Roles_ROLETYPE_ID	roles.ROLETYPE_ID	roletypes.ID	Restrict delete
FK_Roles_RoleVersions_Role_ID	roles_roleversions.Role_ID	roles.ID	Restrict delete
FK_Roles_RoleVersions_versions_ID	roles_roleversions.versions_ID	roleversions.ID	Restrict delete
FK_RoleVersions_CHANGERESPONSIBLE_ID	roleversions.CHANGERESPONSIBLE_ID	organisationentitys.ID	Restrict delete
FK_RoleVersions_Permissions_permissions_ID	roleversions_permissions.permissions_ID	permissions.ID	Restrict delete
FK_RoleVersions_Permissions_RoleVersion_ID	roleversions_permissions.RoleVersion_ID	roleversions.ID	Restrict delete
FK_RoleVersions_ROLE_ID	roleversions.ROLE_ID	roles.ID	Restrict delete
OrganisationEntitys_OrganisationEntityschildren_ID	organisationentitys_organisationentitys.children_ID	organisationentitys.ID	Restrict delete
rgnstionEntitysOrganisationEntitysrgnstionalUnitID	organisationentitys_organisationentitys.OrganisationalUnit_ID	organisationentitys.ID	Restrict delete

24.3 Persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns=http://java.sun.com/xml/ns/persistence
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
<persistence-unit name="rmt" transaction-type="JTA">
<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
<jta-data-source>java:/rmtDS</jta-data-source>

<class>model.Application</class>
<class>model.BaseEntity</class>
<class>model.OrganisationalUnit</class>
<class>model.OrganisationEntity</class>
<class>model.Permission</class>
<class>model.Person</class>
<class>model.Role</class>
<class>model.RoleType</class>
<class>model.RoleVersion</class>

<properties>

      <!-- connection properties -->
      <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/rmtdb"/>

      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="rmt123" />

      <!-- eclipse link properties -->
      <property name="eclipselink.target-server" value="JBoss" />
      <property name="eclipselink.target-database" value="MySQL" />
      <property name="eclipselink.ddl-generation" value="create-or-extend-tables"
/>
      <property name="eclipselink.ddl-generation.output-mode" value="database" />
      <property name="eclipselink.logging.level" value="WARNING" />

      </properties>
</persistence-unit>
</persistence>
```

25. Grössen und Leistung

Mehr Speicher ist immer gut – unsere Applikation kommt auch mit weniger aus. Probleme ergeben sich beim einlesen von grossen .csv Dateien oder beim gleichzeitigen Abrufen von vielen verschiedenen Daten. Auf unseren Übungsrechnern hat das Parsen, welches zugegebenermassen nicht sonderlich performant implementiert ist, bei knapp 28‘000 Zeilen beim Permissions lesen und bei gut 7000 Zeilen beim Rollen lesen ein Timeout ausgelöst, welches nach 5min die Transaktion komplett abbricht und zurückrollt. Bei grösserem Arbeitsspeicher und mehr Rechenleistung dürften diese Werte höher liegen, aber vermutlich müsste man das Timeout deutlich erhöhen oder gar ausschalten, will man über 100'000 Zeilen lange Dateien einlesen. Das beobachtete Leistungsoptimum liegt um die 10‘000 Zeilen pro Lesevorgang, danach sinkt die Lesegeschwindigkeit langsam, aber stetig.
Bei grossen Datenmengen und vielen Aufrufen vieler verschiedener Daten konnte auch eine merkliche, wenn auch keine einschneidende Erhöhung der Seitenladezeit festgestellt werden. Dieser Effekt dürfte auf einer leistungsstärkeren Maschine deutlich schwächer sein, kann aber auch da zu Problemen führen.

Glossar

Änderungsgeschichte

Datum	Version	Änderung	Autor
05.03.2013	1.0	Initialversion	msondere

26. Glossar

Begriff	Beschreibung
RMT	Rollenmanagement-Tool
SA	Semesterarbeit
EJB3	Enterprise Java Beans 3
JPA	Java Persistence API
RUP	Rational Unified Process
JSF	Java Server Faces
AS	Application Server
JVM	Java Virtual Machine
JRE	Java Runtime Environment
DI	Dependency Injection
SVN	Subversion
VM	Virtual Machine